Formal Aspects of Computing (2001) 0: 1–23
© 2001 BCS

A New Approach to Abstract Syntax with Variable Binding[1]

Murdoch J. Gabbay and Andrew M. Pitts
Cambridge University Computer Laboratory, Cambridge, UK

Keywords: Abstract syntax; Alpha-conversion; Permutation actions; Set theory; Structural induction

Abstract. The permutation model of set theory with atoms (FM-sets), devised by Fraenkel and Mostowski in the 1930s, supports notions of 'name-abstraction' and 'fresh name' that provide a new way to represent, compute with, and reason about the syntax of formal systems involving variable-binding operations. Inductively defined FM-sets involving the name-abstraction set former (together with Cartesian product and disjoint union) can correctly encode syntax modulo renaming of bound variables. In this way, the standard theory of algebraic data types can be extended to encompass signatures involving binding operators. In particular, there is an associated notion of structural recursion for defining syntax-manipulating functions (such as capture avoiding substitution, set of free variables, etc.) and a notion of proof by structural induction, both of which remain pleasingly close to informal practice in computer science.

1. Introduction

This paper presents a simple variant of classical set theory that turns out to be very convenient for developing the metamathematics of formal systems involving variable-binding operations. The main message we wish to get across is that a relatively small change to classical set theory, involving the use of *atoms* (or *urelements*, as they are often referred to), can reap great benefits for this aspect of metamathematics. In particular we will see that in such a set theory it is possible to represent α-equivalence classes of abstract syntax trees as the elements of an inductively defined set, rather than just the quotient of such a set. Moreover, this can be done in such a way that the techniques of structural induction and recursion, which Burstall did so much to promote in computer science [Bur69, Bur77], extend in a pleasant fashion from conventional algebraic datatypes to ones involving variable-binding operations.

But of course using a different set theory, even though it is very close to the usual one, is not a step to be undertaken lightly; so we begin with a critique of the current state of the art. It is oriented towards

Correspondence and offprint requests to: Andrew M. Pitts, Cambridge University Computer Laboratory, J. J. Thompson Avenue, Cambridge CB3 0FD, UK. e-mail: Andrew.Pitts@cl.cam.ac.uk

[1] This paper is dedicated to Rod Burstall. It is a revised and expanded version of Gabbay and Pitts [GaP99].

the particular use of such a metatheory of variable binding that most concerns us, namely machine-assisted proofs about the structural operational semantics [Plo81] of programming languages.

1.1. Background

The theory and practice of specifying and reasoning about syntactical structures that *do not* involve variable-binding constructs is well understood. The theory involves such indispensable concepts as algebraic data types [GTWW77], inductively defined sets, definition by structural recursion and proof by structural induction [Bur69]; the practice can be seen in several general-purpose systems for machine-assisted proof (such as [GoM93, PaM93]). This algebraic, 'no binders' machinery is often applied to syntax that *does* involve binders; but in that case it yields overly-concrete representations in which large numbers of essentially routine constructions and proofs to do with renaming bound variables, capture avoiding substitution, and so on, must be done and redone for each object-language on a case-by-case basis. To make large, machine-checkable proofs feasible, one has to take a more sophisticated approach.

One such approach involves representing object-level variables by variables in a metalanguage based on typed λ-calculus. This shifts renaming and substitution to the meta-level where their properties are established once and for all. This is the 'higher order abstract syntax' (HOAS) approach [PfE88] — an idea going back to Church [Chu40] and Martin-Löf [Mar84] that has found its way into many of the current logical frameworks and proof assistants. Its big drawback, in its original form at least, is that one looses the ability to define functions on syntax by structural recursion and to prove properties by structural induction — absolutely essential tools, especially for applications to operational semantics. There are recent proposals to overcome this shortcoming [McM97, DPS97]. They result in systems that are technically very interesting but force the designer of algorithms and proofs 'modulo-α-conversion' to use forms of expression that are rather far from familiar informal practice. Indeed, the whole HOAS approach by its very nature disallows a feature that we regard of key practical importance: *the ability to manipulate names of bound variables explicitly* in computation and proof.[2] The same criticism applies to approaches to variable binding based upon de Bruijn's nameless terms [dB72] or categorical combinators [Cur93]: these are good for machine implementations, but not, we would argue, for representations intended for machine-assisted *human* reasoning.

Instead of the HOAS approach of moving both α-conversion and substitution to the meta-level, we will just promote the former, leaving notions of substitution to be defined (by structural recursion) on a case-by-case basis by the user. This does not seem too bad a compromise, since we show that it permits both a nice calculus of bound names and notions of structural recursion and structural induction for variable-binding constructs. We present these in a theory with the expressive power of classical set theory (without the axiom of choice) and which remains close to informal practice in its forms of expression. This, and the focus on α-conversion rather than substitution as primitive, makes our work close in spirit to that of Gordon and Melham [GoM96], who axiomatise a type of untyped λ-terms modulo α-conversion within Church's higher-order logic. However, we take a more foundational approach, in that the necessary properties of α-conversion become part of the underlying set theory[3] itself. This results in notions of structural recursion and induction that seem rather simpler than those in [GoM96] (cf. Example 6.10 below).

1.2. Overview of the paper

The Fraenkel-Mostowski permutation model of set theory was devised in order to prove the independence of the Axiom of Choice (AC) from the other axioms of Zermelo-Fraenkel set theory *with atoms* (ZFA) (and some decades later Cohen proved the harder result of independence of AC from set theory without atoms (ZF), via his celebrated forcing method; see [Jec77, Section 6] for a brief survey of these matters). Our application of the permutation model of set theory to abstract syntax with variable binding is rather far from this

[2] Of course, one can introduce a type of 'names' in a HOAS-style signature, as is done for example in [HMS98]; but as the authors of that work say (p. 26): 'The main drawback of HOAS is the difficulty of dealing with metatheoretic issues concerning names.... As a consequence, some metatheoretic properties involving substitution and freshness of names...cannot be proved inside the framework and instead have to be postulated.' It is precisely such problems with names that we claim our approach overcomes in a simple way.

[3] Instead of using set theory, one can use a higher-order logic based on Church's simple theory of types [Chu40] as the basis of the approach described here, but we have yet to develop this formulation.

original purpose! To motivate its use, in Section 2 we consider a paradigmatic example of syntax involving variable binding, the lambda calculus, and show that α-conversion of λ-terms can be expressed in terms of the primitive operation of permuting a term's variables, be they free, bound or binding (Proposition 2.2). This motivates the use of sets equipped with permutation actions as a general setting for discussing syntax: the basic notions, including the key notion of 'finite support', are reviewed in Section 3; and the Fraenkel-Mostowski cumulative hierarchy of sets with finite support is recalled in Section 4. That section introduces *FM-set theory*, an axiomatisation of the Fraenkel-Mostowski permutation model whose special features, compared with the usual Zermelo-Fraenkel set theory, are expressed in terms of a set of atoms, a 'freshness' predicate $a \# x$ (meaning 'atom a is not in the (finite) support of object x') and a quantifier $\mathsf{N}a \in \mathsf{A}\,\phi$ (meaning 'for all but finitely many atoms a, ϕ holds'). As far as we know, the N-quantifier is novel; it has a dual 'some/any' character (see Lemma 4.10), which gives it very pleasant logical properties that we exploit throughout the rest of the paper. Our main contribution occurs in Section 5, where we define a new notion of 'atom-abstraction' within FM-set theory. Then in Section 6 we show that atom-abstraction can be used in combination with Cartesian product and disjoint union to form inductively defined sets in the Fraenkel-Mostowski permutation model that represent syntax modulo α-conversion (Theorem 6.2). The standard theory of algebraic data types extends to encompass signatures involving binding operators. In particular, there are notions of structural recursion (Corollary 6.7) and induction (Theorem 6.8) for these inductively defined FM-sets that respect indistinguishability of α-variants and yet are pleasingly close to informal practice when manipulating α-equivalence classes via representatives with explicitly named bound variables. Section 7 discusses related work; and in Section 8, we conclude by mentioning some ways in which the theory presented here is being developed and applied.

2. Permutative Renaming

As a motivating example, consider the terms M of the untyped lambda calculus [Bar84]:

$$M ::= a \mid M\,M \mid \lambda a.M \tag{1}$$

where a ranges over some countably infinite set $\mathbb{A}$ of names of variables.

Definition 2.1. The countably infinite set $\mathbb{A}$ will be fixed throughout this paper. We call its elements *atoms*.

Given $a, a' \in \mathbb{A}$, consider the following three versions of the notion of variable-renaming for λ-terms M:

$\{a'/a\}M$, the textual substitution of a' for all free occurrences of a in M;
$[a'/a]M$, the capture-avoiding substitution of a' for all free occurrences of a in M;
$(a'\,a)\cdot M$, the *transposition* of *all* occurrences (be they free, bound, or binding) of a and a' in M.

For example, when M is $(\lambda a'.a\,a')(\lambda a.a)$, then

$$\begin{aligned}\{a'/a\}M &= (\lambda a'.a'a')(\lambda a.a),\\ [a'/a]M &= (\lambda a''.a'a'')(\lambda a.a) \quad (a'' \text{ fresh}),\\ (a'\,a)\cdot M &= (\lambda a.a'a)(\lambda a'.a').\end{aligned}$$

We assume the reader is familiar with the first two notions of renaming (if not, see [Bar84], for example). Although the third version is possibly less familiar, it is in fact more basic than the other two, because: firstly, one does not need to know whether any of the operations in the underlying signature for λ-terms are supposed to be variable-binders in order to define it; secondly, it can nevertheless be used to define α-conversion, as the following result shows (cf. [Gun92, p. 36], which uses $\{a'/a\}(-)$ in place of $(a'\,a)\cdot(-)$ for the same purpose). Recall that *α-conversion*, $=_\alpha$, is usually defined as the least congruence on the set of λ-terms that identifies $\lambda a.M$ with $\lambda a'.[a'/a]M$.

Proposition 2.2. The relation of α-conversion between λ-terms, $=_\alpha$, coincides with the binary relation $\sim$ inductively generated by the following axioms and rules.

$$\frac{a \in \mathbb{A}}{a \sim a} \qquad \frac{M_1 \sim M_1' \quad M_2 \sim M_2'}{M_1\,M_2 \sim M_1'\,M_2'} \qquad \frac{(a''\,a)\cdot M \sim (a''\,a')\cdot M' \qquad a'' \neq a, a' \qquad a'' \text{ does not occur in } M, M'}{\lambda a.M \sim \lambda a'.M'}$$

Proof. It is not hard to see that $(a'\ a)\cdot(-)$ preserves $=_\alpha$ and hence that $=_\alpha$ is closed under the axioms and rules defining $\sim$. Therefore $\sim$ is contained in $=_\alpha$. The converse follows by proving that $\sim$ is a congruence relating $\lambda a.M$ and $\lambda a'.[a'/a]M$: this follows from the easily verified facts that $(a'\ a)\cdot(-)$ preserves $\sim$, and that if a' does not occur in M, then $(a'\ a)\cdot M \sim [a'/a]M$. □

This proposition shows that matters to do with variable binding can be phrased in terms of the elementary operation of variable-transposition, $(a'\ a)\cdot(-)$, rather than the more complicated operation of variable-substitution, be it textual or capture-avoiding. Both *are* more complicated than transposition because they depend on the auxiliary definition of what are the free variables of a term. Note that transposition is an instance of the more general operation of *permuting the atoms in M according to a bijection $\pi : \mathbb{A} \cong \mathbb{A}$*, the result of which we write as $\pi \cdot M$. This 'permutation action' permits one to formalise an important abstractness property of metatheoretic assertions involving the notion of 'variable', namely that *the validity of assertions about syntactical objects should be sensitive only to* distinctions *between variable names, rather than to the particular names themselves.* Put more formally, this is the *equivariance* property of an assertion $\phi(M)$ about terms M:

$$\forall \pi, M\ (\phi(M) \ \Leftrightarrow\ \phi(\pi \cdot M)) \tag{2}$$

the validity of which of course depends upon the nature of the assertion $\phi(M)$ (in particular one would expect M to be the only free metavariable in ϕ for (2) to be valid; cf. Lemma 4.7). Such notions belong to the rich mathematical theory of sets equipped with a permutation action, which we recall next. It is worth noting that much of that theory would be inapplicable were one to try to base the development upon arbitrary (or even injective) renaming functions from atoms to atoms, rather than upon permutations.

3. Permutation Actions and Finite Support

Let $perm(\mathbb{A})$ denote the group of all permutations of $\mathbb{A}$. Thus the elements π of $perm(\mathbb{A})$ are bijections from $\mathbb{A}$ to itself. The group multiplication takes two such bijections π and π' and composes them — we write the composition of π followed by π' as $\pi' \circ \pi$. The group identity is the identity function on $\mathbb{A}$, denoted by $id_{\mathbb{A}}$. As one may gather from the discussion above, we will be particularly concerned with the elements of $perm(\mathbb{A})$ that are *transpositions*: given $a, b \in \mathbb{A}$, we write $(a\ b)$ for the permutation that interchanges a and b, leaving all other atoms fixed:

$$(a\ b)(c) \ \triangleq\ \begin{cases} a & \text{if } c = b \\ b & \text{if } c = a \\ c & \text{otherwise} \end{cases} \tag{3}$$

Definition 3.1. A *$perm(\mathbb{A})$-set* consists of a set X equipped with an *action* of the group $perm(\mathbb{A})$: this is a function $(-)\cdot_X(-)$ mapping pairs $(\pi, x) \in perm(\mathbb{A}) \times X$ to elements $\pi \cdot_X x$ of X and satisfying

$$\pi' \cdot_X (\pi \cdot_X x) = (\pi' \circ \pi) \cdot_X x \quad \text{and} \quad id_{\mathbb{A}} \cdot_X x = x$$

for all $\pi, \pi' \in perm(\mathbb{A})$ and $x \in X$.

We will usually refer to a $perm(\mathbb{A})$-set via its underlying set when the associated action is clear from the context.

Example 3.2. Of the many examples of sets equipped with an action of the group $perm(\mathbb{A})$, we mention some that will be relevant here.

(i) The set $\mathbb{A}$ itself has a $perm(\mathbb{A})$-action given by applying the permutation, *qua* function, to atoms:

$$\pi \cdot_{\mathbb{A}} a \ \triangleq\ \pi(a).$$

(ii) The set Λ of λ-terms M, defined as in equation (1), possesses a $perm(\mathbb{A})$-action defined by induction on the structure of M by:

$$\begin{cases} \pi \cdot_\Lambda a & \triangleq\ \pi \cdot_{\mathbb{A}} a \\ \pi \cdot_\Lambda (M\ M') & \triangleq\ (\pi \cdot_\Lambda M)(\pi \cdot_\Lambda M') \\ \pi \cdot_\Lambda (\lambda a.M) & \triangleq\ \lambda(\pi \cdot_{\mathbb{A}} a).(\pi \cdot_\Lambda M) \end{cases}$$

(iii) Note that if two λ-terms are α-equivalent, $M =_\alpha M'$, then for any permutation π one also has $\pi \cdot_\Lambda M =_\alpha \pi \cdot_\Lambda M'$. It follows that $(-) \cdot_\Lambda (-)$ induces an action on the quotient set $\Lambda/=_\alpha$:

$$\pi \cdot_{(\Lambda/=_\alpha)} [M]_\alpha \triangleq [\pi \cdot_\Lambda M]_\alpha$$

(where we write $[M]_\alpha$ for the α-equivalence class of the λ-term M).

(iv) If X is a $perm(\mathbb{A})$-set, with action $(-) \cdot_X (-)$, then we can endow its powerset $pow(X) = \{S \mid S \subseteq X\}$ with a $perm(\mathbb{A})$-action by defining:

$$\pi \cdot_{pow(X)} S \triangleq \{\pi \cdot_X x \mid x \in S\}$$

(v) If X and Y are $perm(\mathbb{A})$-sets, then we can endow the disjoint union $X + Y = \{(0, x) \mid x \in X\} \cup \{(1, y) \mid y \in Y\}$ with a $perm(\mathbb{A})$-action by defining:

$$\pi \cdot_{(X+Y)} z \triangleq \begin{cases} (0, \pi \cdot_X x) & \text{if } z = (0, x) \\ (1, \pi \cdot_Y y) & \text{if } z = (1, y) \end{cases}$$

(vi) The unique function $\mathbb{A} \times \emptyset \to \emptyset$ is a $perm(\mathbb{A})$-action on the emptyset $\emptyset$.

(vii) We say that X is a *sub-$perm(\mathbb{A})$-set* of Y if $X \subseteq Y$ and $\pi \cdot_Y (-)$ restricts to a function $X \to X$, for any $\pi \in perm(\mathbb{A})$. In that case X with the restricted action is itself a $perm(\mathbb{A})$-set. If $\mathcal{X}$ is a set of $perm(\mathbb{A})$-sets that is linearly ordered by this relation, then the union $\bigcup \mathcal{X} = \{x \mid \exists X \in \mathcal{X}(x \in X)\}$ has a $perm(\mathbb{A})$-action given by the actions on each $X \in \mathcal{X}$ (which agree where they overlap):

$$\pi \cdot_{(\bigcup \mathcal{X})} x = \pi \cdot_X x \quad \text{if } x \in X \in \mathcal{X}$$

In general, an action of $perm(\mathbb{A})$ on a set X gives us an abstract way of regarding the elements x of X as somehow 'involving atoms from $\mathbb{A}$ in their construction', in as much as the action tells us how permuting atoms changes x — which turns out to be all we need for an abstract theory of variable renaming and binding. An important part of this theory is the notion of *finite support*. This generalises the property of an abstract syntax tree that it only involves finitely many atoms in its construction to the abstract level of an element of any set equipped with a $perm(\mathbb{A})$-action.

Definition 3.3 (Finite support). Given a $perm(\mathbb{A})$-set X and an element $x \in X$, a set of atoms $w \subseteq \mathbb{A}$ is said to *support* x if all permutations $\pi \in perm(\mathbb{A})$ that fix every element of w also fix x:

$$\forall a \in w(\pi(a) = a) \;\Rightarrow\; \pi \cdot_X x = x \tag{4}$$

We say that x is *finitely supported* (in X) if there is some finite subset $w \subseteq \mathbb{A}$ supporting it. The $perm(\mathbb{A})$-set X has the *finite support property* if all its elements are finitely supported.

Proposition 3.4 (The support of an object). If an element x of a $perm(\mathbb{A})$-set X is finitely supported, then there is a least finite subset of $\mathbb{A}$ supporting x. We write $supp(x)$ for this (necessarily unique) finite set of atoms and call it *the support of* x (leaving implicit which $perm(\mathbb{A})$-set X is being referred to).

Proof. We give a proof that emphasises the role of transpositions (see equation (3)) and the predicate 'a is not in the support of x', which play crucial roles in the theory we develop in this paper. Indeed, one can give an explicit formula for $supp(x)$ in terms of the action of transpositions on x:

$$supp(x) \triangleq \{a \in \mathbb{A} \mid \{b \in \mathbb{A} \mid (a\,b) \cdot_X x \neq x\} \text{ is not finite}\} \tag{5}$$

Thus for any $a \in \mathbb{A}$, $a \notin supp(x)$ holds if and only if $(a\,b) \cdot_X x = x$ holds for all but finitely many $b \in \mathbb{A}$. (Cf. the definition of the И-quantifier given below in Definition 4.4.) We show that this definition of $supp(x)$ has the following three properties:

(i) If w is a finite set of atoms supporting x, then $supp(x) \subseteq w$.

(ii) If $a, b \in \mathbb{A} - supp(x)$, then $(a\,b) \cdot_X x = x$.

(iii) If w is a finite set of atoms supporting x, then so is $w - \{a\}$, for any $a \in w - supp(x)$.

So if x is finitely supported, (i) together with finitely many applications of (iii) allows us to deduce that $supp(x)$ is indeed a finite set supporting x and is the least such. We need (ii) in order to prove (iii).

Property (i) follows almost immediately from the definition of $supp(x)$ in equation (5). For suppose w is a finite set of atoms supporting x. If $a, b \notin w$, then $\forall c \in w\,.\,(a\,b)(c) = c$, so since w supports x, we have

$(a\,b)\cdot_X x = x$. Therefore $a \notin w$ implies $\{b \in \mathbb{A} \mid (a\,b)\cdot_X x \neq x\}$ is contained in w and hence is finite, so that $a \notin \mathit{supp}(x)$. Thus $a \in \mathit{supp}(x)$ implies $a \in w$, as required.

To prove property (ii), clearly it suffices to consider the case when $a \neq b$. So suppose a and b are distinct atoms not belonging to $\mathit{supp}(x)$. By definition of $\mathit{supp}(x)$, the set of atoms $\{c \in \mathbb{A} \mid (a\,c)\cdot_X x \neq x\} \cup \{c \in \mathbb{A} \mid (b\,c)\cdot_X x \neq x\} \cup \{a, b\}$ is finite; so we can find an atom c not in this set, i.e. such that $c \neq a, b$ and

$$(a\,c)\cdot_X x = x = (b\,c)\cdot_X x \tag{6}$$

Since a, b, c are all distinct, the permutations $(a\,b)$ and $(a\,c)\circ(b\,c)\circ(a\,c)$ are equal. Therefore from equation (6) we have

$$(a\,b)\cdot_X x = ((a\,c)\circ(b\,c)\circ(a\,c))\cdot_X x = (a\,c)\cdot_X ((b\,c)\cdot_X ((a\,c)\cdot_X x)) = x$$

as required.

Finally, to prove property (iii), suppose w is a finite set of atoms supporting x, that $a \in w - \mathit{supp}(x)$, and that $\pi \in \mathit{perm}(\mathbb{A})$ satisfies $\forall b \in w - \{a\}(\pi(b) = b)$. We have to prove that $\pi \cdot_X x = x$. We may assume $\pi(a) \neq a$, for otherwise π fixes every element of w and hence $\pi \cdot_X x = x$, because x is supported by w. But then we also have $\pi^{-1}(a) \notin w - \{a\}$, since otherwise $\pi^{-1}(a)$, being an element of $w - \{a\}$, would be fixed by π, so that $a = \pi(\pi^{-1}(a)) = \pi^{-1}(a)$, and hence $\pi(a) = a$, contrary to assumption. By property (i) and assumption on a, we have $\mathit{supp}(x) \subseteq w - \{a\}$ and hence $a, \pi^{-1}(a) \notin \mathit{supp}(x)$. Therefore by property (ii), $(\pi^{-1}(a)\ a)\cdot_X x = x$. So

$$\pi \cdot_X x = \pi \cdot_X ((\pi^{-1}(a)\ a)\cdot_X x) = (\pi \circ (\pi^{-1}(a)\ a))\cdot_X x \tag{7}$$

But the permutation $\pi \circ (\pi^{-1}(a)\ a)$ clearly fixes a; and it fixes every element of $w - \{a\}$ because $(\pi^{-1}(a)\ a)$ and π do so. So $\pi \circ (\pi^{-1}(a)\ a)$ fixes all of w and therefore, since w supports x, we have

$$(\pi \circ (\pi^{-1}(a)\ a))\cdot_X x = x \tag{8}$$

Combining equations (7) and (8) yields $\pi \cdot_X x = x$, as required. □

Example 3.5. Consider the $\mathit{perm}(\mathbb{A})$-sets in Example 3.2.

(i) $\mathbb{A}$ has the finite support property, with $\mathit{supp}(a) = \{a\}$ for each $a \in \mathbb{A}$.

(ii) Λ has the finite support property: the support of a λ-term M turns out to be the finite set of atoms occurring in it, whether as free, bound, or binding occurrences.

(iii) The quotient set $\Lambda/{=_\alpha}$ also has the finite support property, with the support of an α-equivalence class of λ-terms being the finite set of atoms occurring *freely* in any representative of the class (it does not matter which).

(iv) A powerset $\mathit{pow}(X)$ does not necessarily have the finite support property even if X does. For example, the only elements of finite support in $\mathit{pow}(\mathbb{A})$ are the subsets $S \subseteq \mathbb{A}$ that are either finite or *cofinite* (i.e. whose relative complement $\mathbb{A} - S$ is finite). However, note that given any $\mathit{perm}(\mathbb{A})$-set X, if $S \in \mathit{pow}(X)$ is supported by a finite set of atoms w, then $\pi \cdot_{\mathit{pow}(X)} S$ is supported by the finite set $\pi \cdot_{\mathit{pow}(\mathbb{A})} w$, for any $\pi \in \mathit{perm}(\mathbb{A})$. Therefore the $\mathit{perm}(\mathbb{A})$-action on $\mathit{pow}(X)$ restricts to one on the subset

$$\mathit{pow}_{\mathrm{fs}}(X) \triangleq \{S \subseteq X \mid S \text{ is finitely supported in } \mathit{pow}(X)\}$$

and $\mathit{pow}_{\mathrm{fs}}(X)$ has the finite support property (whether or not X does).

(v) If X and Y have the finite support property, then so does their disjoint union $X + Y$: each $z \in X + Y$ is either of the form $(0, x)$ and supported by the finite set of atoms supporting x in X, or is of the form $(1, y)$ and supported by the finite set of atoms supporting y in Y.

(vi) The empty set has the finite support property, for rather trivial reasons.

(vii) If each element of a linearly ordered set $\mathcal{X}$ of $\mathit{perm}(\mathbb{A})$-sets has the finite support property, then so does its union: each $x \in \bigcup \mathcal{X}$ is supported by the finite set of atoms that supports it in any $X \in \mathcal{X}$ containing x.

4. FM-Set Theory

The $\mathit{perm}(\mathbb{A})$-sets with the finite support property are the objects of a category whose morphisms $f : X \longrightarrow Y$ are the so-called *equivariant functions* — those functions satisfying $f(\pi \cdot_X x) = \pi \cdot_Y f(x)$ for all $\pi \in \mathit{perm}(\mathbb{A})$

and $x \in X$. At this point we could develop our theory of name-abstraction in terms of constructions within this category: see [PiG00, Appendix]. Instead, here we take a set-theoretic approach and construct a single, 'large' $perm(\mathbb{A})$-set $\mathcal{FM}(\mathbb{A})$ with the finite support property, in which our theory can be expressed using familiar, set-theoretic language. One benefit is that if a particular construction can be expressed in this language, then the action of permutations is inherited from the ambient universe $\mathcal{FM}(\mathbb{A})$ without having to define it explicitly and without having to prove the associated finite support property. (Compare this with the approach to denotational metalanguages initiated by Scott [Sco93] in which questions of domain-theoretic continuity are made implicit.)

Recall the usual von Neumann cumulative hierarchy of sets [Sho77]:

$$\begin{cases} \mathcal{V}_0 & \triangleq \emptyset \\ \mathcal{V}_{\alpha+1} & \triangleq pow(\mathcal{V}_\alpha) \\ \mathcal{V}_\lambda & \triangleq \bigcup_{\alpha \leqslant \lambda} \mathcal{V}_\alpha \qquad (\lambda \text{ a limit ordinal}) \end{cases} \tag{9}$$

More generally, given a set A we can define a cumulative hierarchy of 'sets involving atoms from A':

$$\begin{cases} \mathcal{V}_0(A) & \triangleq \emptyset \\ \mathcal{V}_{\alpha+1}(A) & \triangleq A + pow(\mathcal{V}_\alpha(A)) \\ \mathcal{V}_\lambda(A) & \triangleq \bigcup_{\alpha \leqslant \lambda} \mathcal{V}_\alpha(A) \qquad (\lambda \text{ a limit ordinal}) \end{cases} \tag{10}$$

where $+$ denotes disjoint union of sets and $pow(-)$ the powerset operation (see (v) and (iv) in Example 3.2). We can build the notions of 'permutation action' and 'finite support property' into such a set-theoretic hierarchy by taking A to be the $perm(\mathbb{A})$-set $\mathbb{A}$ of atoms and replacing $pow(-)$ by the $pow_{\mathrm{fs}}(-)$ operation from Example 3.5(iv):

$$\begin{cases} \mathcal{FM}_0(\mathbb{A}) & \triangleq \emptyset \\ \mathcal{FM}_{\alpha+1}(\mathbb{A}) & \triangleq \mathbb{A} + pow_{\mathrm{fs}}(\mathcal{FM}_\alpha(\mathbb{A})) \\ \mathcal{FM}_\lambda(\mathbb{A}) & \triangleq \bigcup_{\alpha \leqslant \lambda} \mathcal{FM}_\alpha(\mathbb{A}) \qquad (\lambda \text{ a limit ordinal}) \end{cases} \tag{11}$$

Because of the properties noted in Example 3.5, each $\mathcal{FM}_\alpha(\mathbb{A})$ is a $perm(\mathbb{A})$-set with the finite support property and $\mathcal{FM}_\alpha(\mathbb{A})$ is a sub-$perm(\mathbb{A})$-set of $\mathcal{FM}_\beta(\mathbb{A})$ when $\alpha \leqslant \beta$ — enabling us to use the union operation at limit ordinals as in Example 3.5(vii). More generally, when we consider the union of all the $\mathcal{FM}_\alpha(\mathbb{A})$ we get one large $perm(\mathbb{A})$-set (i.e. a $perm(\mathbb{A})$-class) in which every element has finite support.

Definition 4.1 (FM-sets). The union of all $\mathcal{FM}_\alpha(\mathbb{A})$ (as α ranges over the ordinals) is called the *Fraenkel-Mostowski universe* (*over the set of atoms* $\mathbb{A}$) and is denoted $\mathcal{FM}(\mathbb{A})$. Using the suggestive names *atm* and *set* for the functions $x \mapsto (0, x)$ and $x \mapsto (1, x)$ giving the two inclusions into a disjoint union, from definition (11) we have that every element x of $\mathcal{FM}(\mathbb{A})$ is either of the form $atm(a)$ with $a \in \mathbb{A}$, or of the form $set(X)$ where X is a finitely supported set of FM-sets formed at an earlier ordinal stage than x (cf. Example 3.5(iv)). We call elements of the form $set(X)$ *FM-sets* and elements of the form $atm(a)$ atoms (a slight abuse of terminology). We will just write $(-) \cdot (-)$ for the $perm(\mathbb{A})$-action on the Fraenkel-Mostowski universe; it satisfies

$$\pi \cdot atm(a) = atm(\pi(a)), \quad \pi \cdot set(X) = set(\{\pi \cdot x \mid x \in X\}) \tag{12}$$

Remark 4.2 (Pure FM-sets). Given an element x of the Fraenkel-Mostowski universe $\mathcal{FM}(\mathbb{A})$, let $TC(x)$ denote the transitive closure of x under the appropriate membership relation, namely $\in_{\mathcal{FM}(\mathbb{A})} = \{(x, y) \mid \exists Y(x \in Y \wedge set(Y) = y\}$. Note that $\mathcal{FM}(\mathbb{A})$ contains a copy of the von Neumann universe $\mathcal{V}$, consisting of those FM-sets x for which $TC(x)$ is disjoint from $\{atm(a) \mid a \in \mathbb{A}\}$. We call such an x a *pure* FM-set.

The usual set-theoretic constructions can be carried out within $\mathcal{FM}(\mathbb{A})$ to build various sets. In particular we will make use of the set ω of natural numbers, and the usual constructions of ordered pairs, Cartesian products, disjoint unions, power- and function-sets. To be more precise and to develop the properties of the Fraenkel-Mostowski universe further, we introduce a suitable theory of atoms and FM-sets within classical first-order logic with equality. This theory is based upon ZFA — *Zermelo-Fraenkel set theory with Atoms* (see for example [Fou80], or [JoM95, §5]). This has a signature containing not only a binary relation symbol

'$\in$' for membership, but also a constant A for the set of atoms, $\mathbb{A}$. The axioms of ZFA are as follows:

(Sets)	$x \in y \Rightarrow y \notin \mathsf{A}$
(Extensionality)	$x \notin \mathsf{A} \wedge y \notin \mathsf{A} \wedge \forall z(z \in x \Leftrightarrow z \in y) \Rightarrow x = y$
(Separation)	$\exists y \notin \mathsf{A}\, \forall z(z \in y \Leftrightarrow (z \in x \wedge \phi))$, y not free in ϕ
($\in$-Induction)	$\forall x(\forall y \in x\, [y/x]\phi \Rightarrow \phi) \Rightarrow \forall x\, \phi$
(Collection)	$\forall y \in x\, \exists w\, \phi \Rightarrow \exists z\, \forall y \in x\, \exists w \in z\, \phi$
(Pairing)	$\exists z(x \in z \wedge y \in z)$
(Union)	$\exists y\, \forall z(z \in y \Leftrightarrow \exists w \in x(z \in w))$
(Powerset)	$\exists y\, \forall z(z \in y \Leftrightarrow \forall w \in z(w \in x))$
(Infinity)	$\exists x(\exists y(y \in x) \wedge \forall y \in x\, \exists w \in x(y \in w))$.

Remark 4.3 (Atoms). Axiom (Sets) expresses the fact that only non-atoms can have elements ('atoms are empty'); the other axioms are like those of the usual Zermelo-Fraenkel set theory, except that certain quantifications $\mathsf{Q}x\,.\,(-)$ (for $\mathsf{Q} = \forall, \exists$) have to be restricted to $\mathsf{Q}x \notin \mathsf{A}\,.\,(-)$ when x must range just over sets rather than over sets and atoms. This difference is mathematically innocuous, but makes it very hard to re-use machine-checked versions of Zermelo-Fraenkel set theory to develop a theory of the Fraenkel-Mostowski universe within a proof assistant. The first author's implementation of such a theory within the Isabelle generic theorem prover [Gab00, Chapter III] gets round this problem by working in a non-well-founded setting and replacing 'empty atoms' by so-called *Quine atoms* [Qui63, Section 4] — objects satisfying $a = \{a\}$ — at the expense of having to alter the axiom of ($\in$-Induction).

The above axioms capture the basic, set-theoretic properties of $\mathcal{FM}(\mathbb{A})$ without saying anything very specific about properties of the set of atoms itself, or of the permutation action. The properties relevant to our intended application depend upon the fact that $\mathbb{A}$ is (countably) infinite and that every element of $\mathcal{FM}(\mathbb{A})$ is finitely supported in the sense of Definition 3.3. We can capture these two properties with the following axioms, which use notation introduced in Definition 4.4:

(A-Not-Finite)	$\mathsf{A} \notin \mathsf{pow}_{\mathrm{fin}}(\mathsf{A})$
(Fresh)	$\exists a \in \mathsf{A}(a \# x)$

Definition 4.4. Here and elsewhere we will freely augment the rather sparse language of ZFA with extensions involving new function and relation symbols (written in this font) uniquely defined by formulas of ZFA. In particular we make the following definitions.

Finite subsets. Define $\mathsf{pow}_{\mathrm{fin}}(x)$ to be a term denoting the set of *finite* subsets of x. One practically useful way of formulating this is to define $\mathsf{pow}_{\mathrm{fin}}(x)$ to be the least set of subsets of x containing the empty set $\emptyset$ and closed under the operations of adding an element of x, $s \mapsto s \cup \{y\}$, for each $y \in x$.

Cofinite subsets. Define $\mathsf{pow}_{\mathrm{cof}}(x)$ to be a term denoting the set of *cofinite* subsets of x, i.e. those $s \subseteq x$ such that the relative complement $x - s$ is in $\mathsf{pow}_{\mathrm{fin}}(x)$.

Permutation action. Define $\mathsf{perm}(\mathsf{A})$ to be a term denoting the set of permutations of A. In view of (12), the action of such permutations $\pi \in \mathsf{perm}(\mathsf{A})$ can be defined by $\in$-recursion as follows:

$$\pi \cdot x \;=\; \begin{cases} \pi(x) & \text{if } x \in \mathsf{A} \\ \{\pi \cdot y \mid y \in x\} & \text{if } x \notin \mathsf{A} \end{cases}$$

In particular, define $(a\,b) \cdot x$ to be a term denoting the result of interchanging $a, b \in \mathsf{A}$ in x.

И-quantifier. For each formula ϕ of the language of ZFA, define $Иa \in \mathsf{A}\, \phi$ ('for all but finitely many atoms a, ϕ') to be the formula $\exists s \in \mathsf{pow}_{\mathrm{cof}}(\mathsf{A})\, \forall a(a \in s \Leftrightarrow \phi)$ expressing that $\{a \in \mathsf{A} \mid \phi\}$ is a cofinite set of atoms.

Freshness relation. Define the formula $a \# x$ ('atom a is *fresh* with respect to x') as follows:

$$a \# x \;\triangleq\; Иb \in \mathsf{A}((a\,b) \cdot x = x) \tag{13}$$

That this is a reasonable definition of 'freshness' may not be immediately apparent; however, the following proposition shows (amongst other things) that $a \# x$ holds if and only if a is not in the support of x, as defined in Proposition 3.4.

Proposition 4.5. The axioms of *FM-set theory* are by definition the axioms of ZFA plus the two axioms (A-Not-Finite) and (Fresh). Modulo the other axioms, axiom (Fresh) is equivalent to asserting that every x has the finite support property of Definition 3.3, suitably formulated in the language of ZFA:

$$\forall x\, \exists a \in \mathsf{A}(a \# x) \;\Leftrightarrow\; \forall x\, \exists w \in \mathsf{pow}_{\mathrm{fin}}(\mathsf{A})\, \forall \pi \in \mathsf{perm}(\mathsf{A})\, (\forall a \in w(\pi(a) = a)) \Rightarrow \pi \cdot x = x \tag{14}$$

In particular the Fraenkel-Mostowski universe $\mathcal{FM}(\mathbb{A})$ over an infinite set $\mathbb{A}$ (Definition 4.1) satisfies these axioms if we interpret the membership relation '$\in$' by the binary relation $\{(x, y) \in \mathcal{FM}(\mathbb{A}) \times \mathcal{FM}(\mathbb{A}) \mid \exists Y(x \in Y \wedge \mathit{set}(Y) = y\}$ and the constant A by the element $\mathit{set}(\{\mathit{atm}(a) \mid a \in \mathsf{A}\}) \in \mathcal{FM}(\mathbb{A})$.

Proof. To prove (14), first note that for any x, if a, b, c are distinct atoms satisfying $(a\,c) \cdot x = x = (b\,c) \cdot x$, then $(a\,b) \cdot x = x$ (see the proof of property (ii) in the proof of Proposition 3.4). From this we can deduce the following two useful facts:

$$a \# x \wedge b \# x \;\Rightarrow\; (a\,b) \cdot x = x \tag{15}$$

$$\exists a \in \mathsf{A}(a \# x) \;\Leftrightarrow\; И a \in \mathsf{A}(a \# x) \tag{16}$$

Note also that if we define $\mathsf{supp}(x)$ as in equation (5), then by definition of $\#$ we have

$$a \# x \;\Leftrightarrow\; a \notin \mathsf{supp}(x) \tag{17}$$

So if the right-hand side of (14) holds, then for each x, the proof of Proposition 3.4 yields that $\mathsf{supp}(x) \in \mathsf{pow}_{\mathrm{fin}}(\mathsf{A})$; and hence from (A-Not-Finite) $\mathsf{supp}(x) \neq \mathsf{A}$, so that by (16) again, $\exists a \in \mathsf{A}(a \# x)$.

Conversely, if $\forall x\, \exists a \in \mathsf{A}(a \# x)$, then for each x, $\mathsf{supp}(x) \in \mathsf{pow}_{\mathrm{fin}}(\mathsf{A})$ by (16) and (17); and by (15) and (17), $(a\,b) \cdot x = x$ for any $a, b \notin \mathsf{supp}(x)$. So any permutation $\pi \in \mathsf{perm}(\mathsf{A})$ that is finite (i.e. with $\{a \in \mathsf{A} \mid \pi(a) \neq a\} \in \mathsf{pow}_{\mathrm{fin}}(\mathsf{A})$) and that fixes each element of $\mathsf{supp}(x)$ satisfies $\pi \cdot x = x$, because such a permutation is expressible as the composition of finitely many transpositions $(a\,b)$ with $a, b \notin \mathsf{supp}(x)$. But axiom (Fresh) implies that *any* $\pi \in \mathsf{perm}(\mathsf{A})$ is finite: for if $a \# \pi$, it is not hard to see that $\pi(a) = a$; so from (16) we have $\pi(a) = a$ for cofinitely many atoms a, so that π is finite. Thus if (Fresh) holds then for each x, $\mathsf{supp}(x)$ defined as in (5) is a finite set of atoms supporting x. Thus we do indeed have the left-to-right implication in (14). □

Remark 4.6 (Failure of the Axiom of Choice). In the literature, careful presentations of the definition of capture-avoiding substitution quite often make use of a choice function for picking out fresh variables: see [Sto88, Section 2], for example. The vague feeling that such concrete choices should be irrelevant crystallises here into the fact that such choice functions are inconsistent with FM-set theory, because its axioms contradict the Axiom of Choice (AC). For example, the axiom (A-Not-Finite) implies that the set of cofinite subsets of A is a set of non-empty sets; but there can be no choice function for it — a diagonalisation argument (using the fact that the graph of any function from $\mathsf{pow}_{\mathrm{fin}}(\mathsf{A})$ to A must be finitely supported) shows this. (Indeed it can be shown that the axiom of *countable* choice is inconsistent with FM-set theory.)

Mechanised proof assistants based on set theory, or on higher-order logic, often include Hilbert's choice operator, $\varepsilon x\,.\,\phi$, to provide anonymous notations for terms defined by formulas (see [LaP99, Section 2.1]). Since the ε-operator can be used to prove AC, we cannot add it to FM-set theory without inconsistency. However, in a fully formal presentation of FM-set theory it would be both consistent and useful to augment the language of FM-set theory with a notation and axioms for forming terms *uniquely* defined by a formula. As mentioned in Definition 4.4, here we proceed in a semi-formal fashion and just introduce new notation on the fly, uniquely defined by formulas. Corollary 4.8 allows us to bound the support of terms involving such new notation by the support of the notation's parameters. It is a consequence of the following equivariance property (cf. equation (2)), which is important in its own right.

Lemma 4.7 (Equivariance property of FM-set theory). Suppose $\phi(\vec{x})$ is a formula of FM-set theory, with free variables contained in the list of distinct variables $\vec{x}$. Then

$$\forall a, b \in \mathsf{A}\, \forall \vec{x}\, (\phi(\vec{x}) \Leftrightarrow \phi((a\,b) \cdot \vec{x})$$

is provable in FM-set theory (where $\phi((a\,b) \cdot \vec{x})$ denotes the simultaneous (capture-avoiding) substitution of $(a\,b) \cdot x_i$ for all free occurrences of x_i in ϕ, as x_i ranges over $\vec{x}$).

Proof. By induction on the structure of ϕ, using the following properties of the constant and relation symbols

of the language:

$$x = y \Rightarrow (a\,b)\cdot x = (a\,b)\cdot y \tag{18}$$
$$x \in y \Rightarrow (a\,b)\cdot x \in (a\,b)\cdot y \tag{19}$$
$$(a\,b)\cdot \mathsf{A} = \mathsf{A} \tag{20}$$

Property (18) is a trivial consequence of the usual properties of equality; and properties (19) and (20) follow from the definition of the permutation action in Definition 4.4. □

Corollary 4.8 (Support of defined notation). Suppose $\phi(\vec{x})$ and $\psi(\vec{x}, y)$ are formulas of FM-set theory, with free variables contained in the indicated lists of variables, and that

$$\phi(\vec{x}) \;\Rightarrow\; \exists! y\, \psi(\vec{x}, y) \tag{21}$$

holds. If we define the term $\mathsf{f}(\vec{x})$ to be the unique y such that $\psi(\vec{x}, y)$, so that $\phi(\vec{x}) \;\Rightarrow\; \forall y(y = \mathsf{f}(\vec{x}) \Leftrightarrow \psi(\vec{x}, y))$, then

$$\phi(\vec{x}) \;\Rightarrow\; \forall a \in \mathsf{A}(a \# \vec{x} \;\Rightarrow\; a \# \mathsf{f}(\vec{x})) \tag{22}$$

(where $a \# \vec{x}$ stands for a conjunction of freshness formulas, one for each variable in the list $\vec{x}$). Hence $\phi(\vec{x}) \;\Rightarrow\; \mathsf{supp}(\mathsf{f}(\vec{x})) \subseteq \mathsf{supp}(x_1) \cup \cdots \cup \mathsf{supp}(x_n)$, if $\vec{x} = x_1, \ldots x_n$.

Proof. Suppose $\phi(\vec{x})$ and $a \# \vec{x}$ hold. We have to prove $a \# y$ for the unique y such that $\psi(\vec{x}, y)$. By axiom (Fresh) we can find $b \in \mathsf{A}$ with $b \# \vec{x}, y$. Since $b \# y$, by Lemma 4.7 we have $a \# (a\,b)\cdot y$. But $(a\,b)\cdot y = y$ by the uniqueness part of property (21): for not only does $\psi(\vec{x}, y)$ hold (by definition of y), but so also does $\psi(\vec{x}, (a\,b)\cdot y)$ (since Lemma 4.7 applied to $\psi(\vec{x}, y)$ gives $\psi((a\,b)\cdot\vec{x}, (a\,b)\cdot y)$ and $(a\,b)\cdot\vec{x} = \vec{x}$ by property (15), because $a, b \# \vec{x}$). So we do indeed have $a \# y$. □

Example 4.9 (Functions). As usual, an FM-set f is a *(partial) function* if it only contains ordered pairs and is single-valued, i.e. if $\mathsf{Fun}(f)$ holds, where

$$\mathsf{Fun}(f) \;\triangleq\; \begin{array}{l}\forall x \in f\, \exists y, z(x = (y, z)) \\ \quad \wedge\, \forall y, z, z'((y, z) \in f \wedge (y, z') \in f \Rightarrow z = z')\end{array}$$

Using (Collection) and (Separation), if $\mathsf{Fun}(f)$ holds then the *domain of definition* of f is the uniquely defined FM-set $\mathsf{dom}(f)$ of first components of elements of f, i.e. $\mathsf{dom}(f) = \{x \mid \exists y((x, y) \in f)\}$. Similarly the *image* of f is $\mathsf{img}(f) = \{y \mid \exists x((x, y) \in f)\}$. If $\mathsf{Fun}(f)$ and $x \in \mathsf{dom}(f)$, then there is a unique y such that $(x, y) \in f$ and in this case, as usual, we define the application notation $f(x)$ to stand for this y. From Corollary 4.8 we have

$$\mathsf{Fun}(f) \;\Rightarrow\; \forall a \in \mathsf{A}(a \# f \;\Rightarrow\; a \# \mathsf{dom}(f) \wedge a \# \mathsf{img}(f)) \tag{23}$$

$$\mathsf{Fun}(f) \wedge x \in \mathsf{dom}(f) \;\Rightarrow\; \forall a \in \mathsf{A}(a \# f \wedge a \# x \;\Rightarrow\; a \# f(x)) \tag{24}$$

While we are about it, let us mention another piece of notation associated with functions. If X and Y are FM-sets, then the *function set* $X \to Y$ is the FM-set defined by: $f \in (X \to Y) \;\Leftrightarrow\; \mathsf{Fun}(f) \wedge \mathsf{dom}(f) = X \wedge \mathsf{img}(f) \subseteq Y$.

In Definition 4.4 we introduced the notation $И a \in \mathsf{A}\, \phi$ to stand for a quantification over all but finitely many atoms. However, the nature of the set of atoms in FM-set theory endows this quantifier with very special properties, as Proposition 4.10 and Corollary 4.11 show.

Proposition 4.10 (И as a 'freshness' quantifier). For any formula ϕ and list of distinct variables $a, \vec{x}$ in the language of FM-set theory, consider the following formulas:

$$\forall a \in \mathsf{A}\,(a \# \vec{x} \Rightarrow \phi) \tag{25}$$
$$И a \in \mathsf{A}\, \phi \tag{26}$$
$$\exists a \in \mathsf{A}\,(a \# \vec{x} \wedge \phi) \tag{27}$$

Then in FM-set theory, (25) $\Rightarrow$ (26) $\Rightarrow$ (27). If the free variables of ϕ are contained in the list $a, \vec{x}$, then also (27) $\Rightarrow$ (25) and hence in this case the three formulas are provably equivalent.

Proof. Note that by axiom (Fresh) and (16) (and the fact that cofinite sets are closed under finite intersection), we have $\{a \in \mathsf{A} \mid a \# \vec{x}\} \in \mathsf{pow}_{\mathrm{cof}}(\mathsf{A})$. So if (25) holds, $\{a \in \mathsf{A} \mid \phi\}$ contains $\{a \in \mathsf{A} \mid a \# \vec{x}\}$ and hence is cofinite, i.e. (26) holds. Similarly, if (26) holds then $\{a \in \mathsf{A} \mid a \# \vec{x}\} \cap \{a \in \mathsf{A} \mid \phi\}$ is the intersection of two cofinite sets, hence is cofinite and hence by (A-Not-Finite) is non-empty: so (27) holds.

Now suppose the free variables of ϕ are contained in $\{a, \vec{x}\}$ and that (27) holds. We use the equivariance property of formulas of FM-set theory given in Lemma 4.7 to deduce from $a \# \vec{x} \wedge \phi$ that for any $b \in \mathsf{A}$

$$b \# (a\,b) \cdot \vec{x} \wedge \phi(b, (a\,b) \cdot \vec{x}) \tag{28}$$

where $\phi(b, (a\,b) \cdot \vec{x})$ denotes the result of substituting b for all free occurrences of a and $(a\,b) \cdot x_i$ for all free occurrences of x_i in ϕ, as x_i ranges over $\vec{x}$. So if $b \# \vec{x}$, since we also have $a \# \vec{x}$, from (15) we get $(a\,b) \cdot \vec{x} = \vec{x}$ and hence (28) implies $\phi(b, \vec{x})$. So (27) does indeed imply (25). □

Corollary 4.11. The quantifier $И a \in \mathsf{A}\,.\,(-)$ commutes with the propositional connectives: for all formulas ϕ and ψ

$$И a \in \mathsf{A}(\phi * \psi) \;\Leftrightarrow\; (И a \in \mathsf{A}\,\phi) * (И a \in \mathsf{A}\,\psi) \quad \text{where } * \text{ is } \wedge,\ \vee,\ \Rightarrow \text{ or } \Leftrightarrow \tag{29}$$

$$И a \in \mathsf{A}(\neg\phi) \;\Leftrightarrow\; \neg(И a \in \mathsf{A}\,\phi) \tag{30}$$

$$И a \in \mathsf{A}\,\mathit{true} \tag{31}$$

are all provable in FM-set theory. Moreover, if all the elements of an FM-set X have empty support (for example, if X is *pure* in the sense of Remark 4.2), then $И a \in \mathsf{A}\,.\,(-)$ commutes with existential and universal quantification over elements of X:

$$X \notin \mathsf{A} \;\wedge\; \forall a \in \mathsf{A}\,\forall x \in X(a \# x) \;\Rightarrow\; (И a \in \mathsf{A}\,\exists x \in X\,\phi \;\Leftrightarrow\; \exists x \in X\,И a \in \mathsf{A}\,\phi) \;\wedge\; (И a \in \mathsf{A}\,\forall x \in X\,\phi \;\Leftrightarrow\; \forall x \in X\,И a \in \mathsf{A}\,\phi). \tag{32}$$

Proof. These properties of the И-quantifier follow easily from its dual ∀-∃ nature established in Proposition 4.10. (For property (32), we also need to note that if $X \notin \mathsf{A} \wedge \forall a \in \mathsf{A}\,\forall x \in X(a \# x)$ holds, then so does $\forall a \in \mathsf{A}(a \# X)$.) □

In view of Proposition 4.10, we are justified in reading $И a \in \mathsf{A}\,.\,\phi$ as 'for some/any fresh atom a, it is the case that ϕ'. This dual ∃-∀ flavour of the И-quantifier seems to exactly fit many situations where a statement about freshness of variables is required: we choose *some* fresh variable with a particular property, but later on may need the fact that *any* such variable will do. For example, when reasoning about α-conversion of λ-terms using the rules in Proposition 2.2 in a 'top-down' fashion, we satisfy the side-condition in the third rule by picking *some* fresh a''; and when using them in a 'bottom-up' fashion, it is useful to know that *any* fresh a'' will do. We will see further examples of this phenomenon in subsequent sections, where we put the И-quantifier to work.

5. Abstraction Sets

In Proposition 2.2 we saw that α-conversion of λ-terms can be formulated in terms of the operation of transposing two variables in a term, together with the relation that a variable does not occur in a term. In view of Example 3.5(ii), equation (17) and Proposition 4.10, we can re-express the rule for α-conversion of λ-abstractions in Proposition 2.2 as

$$\frac{И a''((a''\,a) \cdot t \sim (a''\,a') \cdot t')}{\lambda a.t \sim \lambda a'.t'}$$

This suggests how to generalise the notion of α-conversion from the syntax trees of λ-terms to arbitrary objects in the Fraenkel-Mostowski universe $\mathcal{FM}(\mathbb{A})$ introduced by Definition 4.1. Let $\sim_{\mathsf{A}}$ be the binary relation on $\mathbb{A} \times \mathcal{FM}(\mathbb{A})$ defined by

$$(a, x) \sim_{\mathsf{A}} (b, y) \;\stackrel{\triangle}{\Leftrightarrow}\; a \in \mathsf{A} \wedge b \in \mathsf{A} \wedge И c \in \mathsf{A}((c\,a) \cdot x = (c\,b) \cdot y) \tag{33}$$

It is not hard to see that $\sim_{\mathsf{A}}$ is an equivalence relation. The following lemma shows that its equivalence classes are FM-sets.

Lemma 5.1 (Atom-abstractions). FM-set theory satisfies

$$(a,x) \sim_{\mathsf{A}} (b,y) \;\Leftrightarrow\; a \in \mathsf{A} \wedge b \in \mathsf{A} \wedge y = (b\,a)\cdot x \wedge (b = a \vee b \# x) \tag{34}$$

Hence by the axiom (Collection), for each $a \in \mathbb{A}$ and $x \in \mathcal{FM}(\mathbb{A})$ there is an FM-set

$$a.x \triangleq \{(b,(b\,a)\cdot x) \mid b \in \mathsf{A} \;\wedge\; (b = a \;\vee\; b \# x)\} \tag{35}$$

which is the $\sim_{\mathsf{A}}$-equivalence class of the pair (a,x). We call $a.x$ the *atom-abstraction* determined by a and x.

Proof. We split the proof of (34) into two cases, according to whether $b = a$ or not. In the first case, the result follows from the easily verified implication

$$(a,x) \sim_{\mathsf{A}} (a,y) \;\Rightarrow\; x = y \tag{36}$$

So suppose $b \neq a$. If $(a,x) \sim_{\mathsf{A}} (b,y)$, then there is some $c \in \mathsf{A}$ with $c \# a,x,b,y$ and

$$(c\,a)\cdot x = (c\,b)\cdot y \tag{37}$$

Since $c \# y$, by the equivariance property (Lemma 4.7) we also have $b = (c\,b)\cdot c \# (c\,b)\cdot y$ and hence by (37), $b \# (c\,a)\cdot x$. From this we get $(c\,a)\cdot b \# (c\,a)\cdot((c\,a)\cdot x) = x$, i.e. $b \# x$ (since $(c\,a)$ leaves b fixed). So for the left-to-right implication in (34), it just remains to show that $y = (b\,a)\cdot x$. But

$$\begin{aligned} y = (c\,b)\cdot((c\,b)\cdot y) &= (c\,b)\cdot((c\,a)\cdot x) && \text{by (37)} \\ &= (b\,a)\cdot((c\,b)\cdot x) && \text{since } (c\,b)\circ(c\,a) = (b\,a)\circ(c\,b) \\ &= (b\,a)\cdot x && \text{by (15) applied to } c \# x \wedge b \# x. \end{aligned}$$

Conversely, if $y = (b\,a)\cdot x$ and $b \# x$, then by axiom (Fresh) there is some $c \in \mathsf{A}$ with $c \# a,x,b,y$ and

$$\begin{aligned} (c\,b)\cdot y &= (c\,b)\cdot((b\,a)\cdot x) && \text{since } y = (b\,a)\cdot x \\ &= (c\,a)\cdot((c\,b)\cdot x) && \text{since } (c\,b)\circ(b\,a) = (c\,a)\circ(c\,b) \\ &= (c\,a)\cdot x && \text{since } c \# x \wedge b \# x \end{aligned}$$

Therefore $И c \in \mathsf{A}((c\,a)\cdot x = (c\,b)\cdot y)$, i.e. $(a,x) \sim_{\mathsf{A}} (b,y)$, as required. □

Corollary 5.2. FM-set theory satisfies

$$a \# a.x \tag{38}$$

$$b \neq a \;\wedge\; b \# x \;\Rightarrow\; b \# a.x \tag{39}$$

$$b \neq a \;\wedge\; b \# a.x \;\Rightarrow\; b \# x \tag{40}$$

Hence $\forall b \in \mathsf{A}(b \# a.x \;\Leftrightarrow\; (b = a \;\vee\; b \# x))$ and therefore $\mathsf{supp}(a.x) = \mathsf{supp}(x) - \{a\}$.

Proof. First note that as a consequence of the equivariance property of Lemma 4.7 one has

$$(c\,b)\cdot(a.x) = ((c\,b)(a)).((c\,b)\cdot x) \tag{41}$$

So if $b \# a,x$, then $(b\,a)\cdot(a.x) = b.((b\,a)\cdot x) = a.x$ by (34); thus $И b \in \mathsf{A}((b\,a)\cdot(a.x) = a.x)$ holds, which is (38). Property (39) follows from property (22) and the easily verified fact that $b \neq a \Leftrightarrow b \# a$. Finally for (40), if $b \neq a$ and $b \# a.x$, i.e. $И c \in \mathsf{A}((c\,b)\cdot(a.x) = a.x)$, then by (41) we have $И c \in \mathsf{A}(a.((c\,b)\cdot x) = a.x)$; hence by (36), $И c \in \mathsf{A}((c\,b)\cdot x = x)$, as required. □

The definition of atom-abstraction embodies a form of 'abstraction as information hiding', since $a.x$ acts like a pair (a,x) in which the name in the first component has been hidden, or made anonymous. However, atom-abstractions also embody a notion of 'abstraction as function', since (36) shows that each atom-abstraction $a.x$ is a partial function; and by construction, the domain of definition of this partial function is a set of atoms. In other words, using the notation of Example 4.9, $\mathsf{Fun}(a.x) \wedge \mathsf{dom}(a.x) \subseteq \mathsf{A}$ holds. Indeed, by (35) and Corollary 5.2

$$\mathsf{dom}(a.x) = \{b \in \mathsf{A} \mid b = a \vee b \# x\} = \mathsf{A} - \mathsf{supp}(a.x) \tag{42}$$

We will use the following special notation for the result of applying such a partial function to an atom in its domain of definition.

Definition 5.3 (Concretion of atom-abstractions). The result of applying an atom-abstraction y, regarded as a partial function, to an atom $b \in \mathsf{dom}(y) = \mathsf{A} - \mathsf{supp}(y)$ will be called the *concretion of y at b* and written $y@b$. From (35) we have that if $y = a.x$, then following form of β-conversion holds for atom-abstraction/concretion:

$$\forall b \in \mathsf{A}((b = a \vee b \# x) \;\Rightarrow\; (a.x)@b = (b\,a) \cdot x) \tag{43}$$

Definition 5.4 (Abstraction sets). For each FM-set X, we can use the (Collection) axiom to deduce the existence of an FM-set $[\mathsf{A}]X$ of *atom-abstractions of elements of X*, defined by:

$$y \in [\mathsf{A}]X \;\stackrel{\Delta}{\Leftrightarrow}\; \exists a \in \mathsf{A}\, \exists x \in X(y = a.x \wedge a \# X) \tag{44}$$

One might wonder why the condition '$a \# X$' is included in the above definition. It is needed to ensure that the concretion of an element of $[\mathsf{A}]X$ at a fresh atom lands us back in X:

$$\forall y \in [\mathsf{A}]X\; \mathit{И} b \in \mathsf{A}(y@b \in X) \tag{45}$$

For note that because of the condition $a \# X$ in (44), plus the fact that $a \# y$ holds when $y = a.x$ (by Corollary 5.2), from Proposition 4.10 we have

$$y \in [\mathsf{A}]X \;\Leftrightarrow\; \mathit{И} a \in \mathsf{A}\, \exists x \in X(y = a.x) \tag{46}$$

Hence given $y \in [\mathsf{A}]X$ and $b \# X, y$, then $y = a.x$ for some $a \in \mathsf{A}$ and $x \in X$ with $a \# b, X$. By (43), $y@b = (b\,a) \cdot x$; and since $x \in X$, we have $(b\,a) \cdot x \in (b\,a) \cdot X$. But $(b\,a) \cdot X = X$, since $a \# X$ and $b \# X$; and hence $y@b \in X$, as required for (45).

Proposition 5.5 (Extensionality for atom-abstractions). A form of η-conversion holds for atom-abstraction/concretion, namely:

$$\forall y \in [\mathsf{A}]X\; \mathit{И} b \in \mathsf{A}(b.(y@b) = y) \tag{47}$$

and hence the following extensionality principle holds:

$$\forall y, y' \in [\mathsf{A}]X(y = y' \;\Leftrightarrow\; \mathit{И} b \in \mathsf{A}(y@b = y'@b)) \tag{48}$$

Proof. Given $y \in [\mathsf{A}]X$ and $b \# y$, by (44) we can find $a \in \mathsf{A}$ and $x \in X$ with $y = a.x$ (and $a \# X$). Then

$$\begin{aligned} b.(y@b) &= b.((b\,a) \cdot x) && \text{since } y@b = (b\,a) \cdot x \text{ by (43)} \\ &= a.x && \text{by (34)} \\ &= y \end{aligned}$$

Thus (47) holds by Proposition 4.10. □

Remark 5.6. The abstraction-set former $[\mathsf{A}](-)$ is remarkably well behaved. In particular it preserves Cartesian products, disjoint unions and even function-sets, up to natural bijections:

$$\begin{aligned} [\mathsf{A}](X \times Y) &\cong ([\mathsf{A}]X) \times ([\mathsf{A}]Y) \\ [\mathsf{A}](X + Y) &\cong ([\mathsf{A}]X) + ([\mathsf{A}]Y) \\ [\mathsf{A}](X \to Y) &\cong ([\mathsf{A}]X) \to ([\mathsf{A}]Y) \end{aligned}$$

For proofs of these bijections, see [Gab00, Corollary 9.6.9].

6. Inductively Defined FM-Sets

We can use the $[\mathsf{A}](-)$ construct in combination with Cartesian product and disjoint union to form inductively defined FM-sets that allow us to view sets of syntax involving binders *modulo α-conversion* as algebraic data types in $\mathcal{FM}(\mathbb{A})$, generalising the 'classical' theory for syntax without binders. To see this, we first need to recall a little of the theory of inductively defined sets, within the context of FM-set theory.

Suppose F is a definable function from FM-sets to FM-sets. In other words, suppose that there is a formula $\phi(X, Y)$ of FM-set theory with the indicated free variables and satisfying $\forall X \notin \mathsf{A}\, \exists! Y \notin \mathsf{A}\, \phi(X, Y)$;

and suppose that for each X, $F(X)$ is the unique Y such that $\phi(X,Y)$. Suppose further that the function F is monotone for inclusions

$$X \notin \mathsf{A} \wedge Y \notin \mathsf{A} \wedge X \subseteq Y \;\Rightarrow\; F(X) \subseteq F(Y) \tag{49}$$

and preserves unions of countable ascending chains

$$\forall n \in \omega(X(n) \notin \mathsf{A} \wedge X(n) \subseteq X(n+1)) \;\Rightarrow\; F(\bigcup\{X(n) \mid n \in \omega\}) = \bigcup\{F(X(n)) \mid n \in \omega\} \tag{50}$$

(Just as we require F to be definable in FM-set theory, so we only consider countable chains of FM-sets $(X(n) \mid n \in \omega)$ that are definable in FM-set theory; so in particular there is one finite set of atoms supporting every $X(n)$, rather than one such set for each $n \in \omega$.) Then F possesses a least (pre)fixed point, $\mu(F)$:

$$F(\mu(F)) = \mu(F) \tag{51}$$

$$F(X) \subseteq X \;\Rightarrow\; \mu(F) \subseteq X \tag{52}$$

We call $\mu(F)$ the *inductively defined FM-set determined by* F. It can be constructed via the familiar Tarski formula:

$$\mu(F) \;\triangleq\; \bigcup\{F^n(\emptyset) \mid n \in \omega\} \qquad \text{where } \begin{cases} F^0(\emptyset) & \triangleq \emptyset \\ F^{n+1}(\emptyset) & \triangleq F(F^n(\emptyset)) \end{cases} \tag{53}$$

The inductively defined FM-sets in which we are interested are all determined by functions F that are also the object part of *functors preserving inclusions*. In other words, not only is F a definable monotone function on FM-sets preserving definable countable chains, but also there is another definable function (traditionally also denoted by F) acting on functions between FM-sets:

$$f \in (X \to Y) \;\Rightarrow\; F(f) \in (F(X) \to F(Y))$$

that preserves composition:

$$f \in (X \to Y) \wedge g \in (Y \to Z) \;\Rightarrow\; F(g \circ f) = F(g) \circ F(f)$$

and inclusion functions:

$$X \subseteq Y \;\Rightarrow\; F(i_{X,Y}) = i_{F(X),F(Y)}$$

where $i_{X,Y} \in (X \to Y)$ is $\{(x,x) \mid x \in X\}$. In this case $\mu(F)$ is the *initial algebra* for the functor F: for every FM-set X and function $f \in F(X) \to X$, there is a unique $\overline{f} \in (\mu(F)) \to X$ such that

$$\begin{array}{ccc} F(\mu(F)) & = & \mu(F) \\ {\scriptstyle F(\overline{f})}\downarrow & & \downarrow{\scriptstyle \overline{f}} \\ F(X) & \xrightarrow[f]{} & X \end{array} \tag{54}$$

commutes (i.e. $\overline{f} = f \circ F(\overline{f})$). Indeed, paralleling the construction of $\mu(F)$ in (53), we have

$$\overline{f} \;\triangleq\; \bigcup\{\overline{f}^n \in (F^n(\emptyset) \to X) \mid n \in \omega\} \quad \text{where } \begin{cases} \overline{f}^0 & \triangleq \emptyset \\ \overline{f}^{n+1} & \triangleq f \circ F(\overline{f}^n) \end{cases} \tag{55}$$

Lemma 6.1. The abstraction-set operation $[\mathsf{A}](-)$ of Definition 5.4 is monotone, preserves unions of definable countable ascending chains of FM-sets and is the object part of a functor preserving inclusions.

Proof. The fact that $[\mathsf{A}](-)$ preserves inclusions and unions of definable countable chains follows immediately from the characterisation of $[\mathsf{A}]X$ in (46) together with the fact (Corollary 4.11) that $И a \in \mathsf{A}(-)$ preserves implications and commutes with $\exists n \in \omega(-)$ (because ω is pure). We extend it to a functor on FM-sets as follows. Given $f \in (X \to Y)$, define

$$[\mathsf{A}]f \;\triangleq\; \{(u,v) \in ([\mathsf{A}]X) \times ([\mathsf{A}]Y) \mid И a \in \mathsf{A}(f(u@a) = v@a)\}$$

That this does give an element of $([\mathsf{A}]X) \to ([\mathsf{A}]Y)$ and that $f \mapsto [\mathsf{A}]f$ preserves composition and inclusion functions all follow from the properties of atom-abstractions established in Section 5 (see [Gab00, Section 10] for more details). □

It is well known that set-functions F built up from constants and projections using the operations of Cartesian product, $(-)\times(-)$, and disjoint union, $(-)+(-)$, enjoy the above properties of monotonicity, preservation of unions of chains and functoriality; and hence such set-functions can be used to form inductively defined sets $\mu(F)$ with the above initial algebra property. This is the starting point for the initial algebra semantics of algebraic signatures [GTWW77] within the usual ZF set theory and its model, the von Neumann hierarchy $\mathcal{V}$ defined by equation (9). All this remains unchanged if we move from $\mathcal{V}$ to the larger (cf. Remark 4.2) Fraenkel-Mostowski cumulative hierarchy $\mathcal{FM}(\mathbb{A})$ and replace ZF by FM-set theory. But now the above lemma shows that within this larger setting of FM-sets we can also use the abstraction-set operation $[\mathsf{A}](-)$ in the construction of F and still be able to form the inductive FM-set $\mu(F)$ as in (53), with its initial algebra property (54). In this way we are able to extend initial algebra semantics to deal with signatures with binding and in particular to get a purely inductive representation of the quotient modulo α-equivalence of the set of abstract syntax trees over such a signature. Rather than treat the case of arbitrary 'binding signatures' (for which see [Gab00, Section 10.2]), we will illustrate these ideas with respect to the paradigmatic example, the signature for the untyped λ-calculus.

Theorem 6.2 ($\Lambda/{=_\alpha}$ as an inductive FM-set). Consider the FM-set Λ of untyped λ-terms, given as the inductively defined FM-set $\mu(F)$ where

$$F(X) \triangleq \mathsf{A} + (X \times X) + (\mathsf{A} \times X) \tag{56}$$

Then the FM-set of equivalence classes of λ-terms modulo the equivalence relation of α-conversion, $\Lambda/{=_\alpha}$, is in bijection with the inductively defined FM-set $\Lambda_\alpha = \mu(F_\alpha)$, where

$$F_\alpha(X) \triangleq \mathsf{A} + (X \times X) + ([\mathsf{A}]X) \tag{57}$$

Proof. This follows by combining the characterisation of $=_\alpha$ given by Proposition 2.2 with the definition of atom-abstraction in Section 5 in terms of the equivalence relation $\sim_{\mathsf{A}}$ given by equation (33). □

In passing from $\Lambda/{=_\alpha}$ to the bijectively equivalent set Λ_α, we replace the global use of quotienting by $=_\alpha$ with local, inductive use of quotienting by $\sim_{\mathsf{A}}$. This is a simplifying step because the quotient sets $[\mathsf{A}]X$ (they are the quotient of $\{(a,x) \in \mathsf{A}\times X \mid a \# X\}$ by $\sim_{\mathsf{A}}$) have some very pleasant properties. In particular, the following lemma gives a simple characterisation of functions of atom-abstractions $a.x \in [\mathsf{A}]X$ in terms of functions of pairs $(a,x) \in \mathsf{A}\times X$.

Lemma 6.3. For all functions $f \in (\mathsf{A}\times X) \to Y$ in $\mathcal{FM}(\mathbb{A})$, there is a unique function $\hat{f} \in ([\mathsf{A}]X) \to Y$ satisfying

$$И a \in \mathsf{A}\, \forall x \in X (\hat{f}(a.x) = f(a,x)) \tag{58}$$

if and only if f satisfies

$$И a \in \mathsf{A}\, \forall x \in X (a \# f(a,x)) \tag{59}$$

Proof. First note that if such an $\hat{f}$ exists, then applying Proposition 4.10 to (58), there is some $a \in \mathsf{A}$ with

$$a \# X, \hat{f}, f \ \wedge\ \forall x \in X (\hat{f}(a.x) = f(a,x))$$

But for any $x \in X$ we always have $a \# a.x$ (Corollary 5.2); and $a \# \hat{f}$ holds by assumption on a; hence by property (24), $a \# \hat{f}(a.x)$, i.e. $a \# f(a,x)$. Thus a satisfies

$$a \# X, f \ \wedge\ \forall x \in X (a \# f(a,x))$$

which by Proposition 4.10 again, is condition (59). There can be at most one function $\hat{f}$ satisfying property (58), because by (44) we have that $(a,x) \mapsto a.x$ is a surjection from $\{(a,x) \in \mathsf{A}\times X \mid a \# X\}$ onto $[\mathsf{A}]X$.

Conversely, suppose condition (59) does hold. Thus by Proposition 4.10 we have

$$\forall a \in \mathsf{A} (a \# X, f \ \Rightarrow\ \forall x \in X (a \# f(a,x))) \tag{60}$$

Define an FM-set $\hat{f}$ by

$$\hat{f} \triangleq \{(z,y) \in ([\mathsf{A}]X) \times Y \mid И a \in \mathsf{A}\, \exists x \in X (z = a.x \wedge f(a,x) = y)\}$$

This is a single-valued relation, because if $(z,y) \in \hat{f}$ and $(z,y') \in \hat{f}$, then by definition of $\hat{f}$, and using axiom (Fresh), we can find some $a \# X, f, z, y, y'$ and $x, x' \in X$ with $z = a.x$, $z = a.x'$, $y = f(a,x)$ and $y = f(a,x')$; then $x = x'$ by property (36) and hence $y = y'$. Moreover, for each $z \in [\mathsf{A}]X$ there is some $y \in Y$ satisfying $(z,y) \in \hat{f}$: for, picking $a \# X, z, f$ and taking x to be the concretion $z@a$ (Definition 5.3), we have $x \in X$ by property (45) and hence by property (60), $y = f(a,x) \in Y$ satisfies $a \# y$. Thus we have

$$\exists a \in \mathsf{A}(a \# X, z, f, y \ \wedge \ \exists x \in X(z = a.x \wedge f(a,x) = y)) \tag{61}$$

so that $(z,y) \in \hat{f}$ (by Proposition 4.10). So $\hat{f}$ is indeed an element of $([\mathsf{A}]X) \to Y$. Furthermore, property (58) holds of it: for if $a \# X, f$, then for any $x \in X$ property (61) holds with $z = a.x$ and $y = f(a,x)$ (since in that case $a \# z$ by (38) and $a \# y$ by (59)), so that $(z,y) \in \hat{f}$, i.e. $\hat{f}(a.x) = \hat{f}(z) = y = f(a,x)$, as required. □

Remark 6.4 (Locally fresh atoms). Given $f \in (\mathsf{A} \times X) \to Y$ satisfying property (59) and given $z \in [\mathsf{A}]X$, we use the notation

$$\mathsf{case}\ z\ \mathsf{of}\ \{a.x \to f(a,x)\} \tag{62}$$

(where the variables a and x are bound in this expression) for the element $\hat{f}(z) \in Y$ specified by the above lemma. From property (58) we have that this 'eliminator' for the abstraction-set construct satisfies the conversion law

$$\forall a' \in \mathsf{A}\, \forall x' \in X\, И b \in \mathsf{A}(\mathsf{case}\ a'.x'\ \mathsf{of}\ \{a.x \to f(a,x)\} = f(b, (b\, a') \cdot x')) \tag{63}$$

As a special case of this construct, given by taking $X = 1 = \{0\}$ in Lemma 6.3, we have that for each $f \in \mathsf{A} \to Y$ satisfying $И a \in \mathsf{A}(a \# f(a))$, there is a unique element of Y, that we denote by

$$\mathsf{new}\ a \in \mathsf{A}\ \mathsf{in}\ f(a) \tag{64}$$

(once again, a is a bound variable in this expression), satisfying

$$И a' \in \mathsf{A}(\mathsf{new}\ a \in \mathsf{A}\ \mathsf{in}\ f(a) = f(a')) \tag{65}$$

In effect, the expression $\mathsf{new}\ a \in \mathsf{A}\ \mathsf{in}\ f(a)$ denotes the use of a 'locally fresh' atom a in the expression $f(a)$; and this is meaningful if the value of $f(a)$ is independent of the choice of a, in the sense that $a \# f(a)$ holds for some/any fresh atom a.

For example, taking $f \in \mathsf{A} \to ([\mathsf{A}]\mathsf{A})$ to be $a \mapsto a.a$, by property (38) we have $\forall a \in \mathsf{A}(a \# a.a)$ and hence there is an element $(\mathsf{new}\ a \in \mathsf{A}\ \mathsf{in}\ a.a) \in [\mathsf{A}]\mathsf{A}$. In fact there is a bijection $[\mathsf{A}]\mathsf{A} \cong 1 + \mathsf{A} \triangleq \{(0,0)\} \cup \{(1,b) \mid b \in \mathsf{A}\}$ given in one direction by

$$\begin{array}{rcl} [\mathsf{A}]\mathsf{A} & \to & 1 + A \\ z & \mapsto & \mathsf{case}\ z\ \mathsf{of}\ \{a.b \to \mathsf{if}\ a = b\ \mathsf{then}\ (0,0)\ \mathsf{else}\ (1,b)\} \end{array}$$

and in the other by

$$\begin{array}{rcl} 1 + A & \to & [\mathsf{A}]\mathsf{A} \\ (0,0) & \mapsto & \mathsf{new}\ a \in \mathsf{A}\ \mathsf{in}\ a.a \\ (1,b) & \mapsto & \mathsf{new}\ a \in \mathsf{A}\ \mathsf{in}\ a.b \end{array}$$

Expression-formers such as $\mathsf{case}\ (-)\ \mathsf{of}\ \{a.x \to (-)\}$ and $\mathsf{new}\ a \in \mathsf{A}\ \mathsf{in}\ (-)$ are part of a term-calculus for atom-abstractions that forms part of the functional programming language proposed by the authors in [PiG00].

Returning to the inductively defined FM-set $\Lambda_\alpha = \mu(F_\alpha)$ introduced in Theorem 6.2, from property (51) we have that Λ_α is a disjoint union of three components

$$\Lambda_\alpha = \mathsf{A} + (\Lambda_\alpha \times \Lambda_\alpha) + ([\mathsf{A}]\Lambda_\alpha)$$

Let

$$\mathsf{var} \in \mathsf{A} \to \Lambda_\alpha \qquad \mathsf{app} \in (\Lambda_\alpha \times \Lambda_\alpha) \to \Lambda_\alpha \qquad \mathsf{lam} \in ([\mathsf{A}]\Lambda_\alpha) \to \Lambda_\alpha$$

denote the inclusion functions of the components into the disjoint union. Combining the least fixed point

and initial algebra properties of $\Lambda_\alpha = \mu(F_\alpha)$ with Lemma 6.3, we obtain the following principles of structural iteration, recursion and induction for our inductive FM-set representing λ-terms modulo α-conversion. (Compare them with the Recursion Scheme of [GoM96, Section 3.1].)

Theorem 6.5 (Structural iteration for Λ_α). Given functions $f \in \mathsf{A} \to X$, $g \in (X \times X) \to X$, and $h \in (\mathsf{A} \times X) \to X$ in $\mathcal{FM}(\mathbb{A})$ with h satisfying

$$\mathit{И}a \in \mathsf{A}\, \forall x \in X (a \# h(a,x)) \tag{66}$$

there is a unique $k \in \Lambda_\alpha \to X$ such that

$$\begin{array}{ll} & \forall a \in \mathsf{A}(k(\mathsf{var}(a)) = f(a)) \\ \wedge & \forall t \in \Lambda_\alpha \forall t' \in \Lambda_\alpha (k(\mathsf{app}(t,t')) = g(k(t),k(t'))) \\ \wedge & \mathit{И}a \in \mathsf{A} \forall t \in \Lambda_\alpha (k(\mathsf{lam}(a.t)) = h(a,k(t))) \end{array} \tag{67}$$

Proof. By Lemma 6.3, the condition (66) means that it induces a function $\hat{h} \in ([\mathsf{A}]X) \to X$. Then f, g and $\hat{h}$ fit together to give a function from the coproduct $\mathsf{A} + (X \times X) + ([\mathsf{A}]X)$ to X, i.e. a function $F_\alpha(X) \to X$. Then k is the unique function $\mu(F_\alpha) \to X$ given by the initial algebra property (54). □

Remark 6.6. Note that this structural iteration principle for Λ_α proves the existence of *total* functions out of Λ_α even though in property (67), as far as λ-abstractions are concerned, we only have to specify what the function does for *fresh* bound variables. Indeed, we can slightly strengthen Theorem 6.5 by only requiring h to be a partial function with $\mathsf{dom}(h) = \{(a,x) \in \mathsf{A} \times X \mid a \# X\}$, for which condition (66) still makes sense. However, if $\mathsf{supp}(X) = \emptyset$, as is the case in the examples below, then $\mathsf{dom}(h)$ is in fact the whole of $\mathsf{A} \times X$.

Corollary 6.7 (Structural recursion for Λ_α). Given functions $f \in \mathsf{A} \to X$, $g \in (\Lambda_\alpha \times \Lambda_\alpha \times X \times X) \to X$, and $h \in (\Lambda_\alpha \times \mathsf{A} \times X) \to X$ in $\mathcal{FM}(\mathbb{A})$ with h satisfying

$$\mathit{И}a \in \mathsf{A}\, \forall t \in \Lambda_\alpha\, \forall x \in X (a \# h(t,a,x)) \tag{68}$$

there is a unique $k \in \Lambda_\alpha \to X$ such that

$$\begin{array}{ll} & \forall a \in \mathsf{A}(k(\mathsf{var}(a)) = f(a)) \\ \wedge & \forall t \in \Lambda_\alpha \forall t' \in \Lambda_\alpha (k(\mathsf{app}(t,t')) = g(t,t',k(t),k(t'))) \\ \wedge & \mathit{И}a \in \mathsf{A} \forall t \in \Lambda_\alpha (k(\mathsf{lam}(a.t)) = h(t,a,k(t))) \end{array} \tag{69}$$

Proof. The passage from iteration to (primitive) recursion for initial algebras uses a standard trick relying upon the uniqueness part of the initial algebra property: define X' to be $\Lambda_\alpha \times X$ and consider $f' \in \mathsf{A} \to X'$, $g \in (X' \times X') \to X'$, and $h' \in (\mathsf{A} \times X') \to X'$ given by

$$\begin{aligned} f'(a) &\triangleq (\mathsf{var}(a), f(a)) \\ g'((t,x),(t',x')) &\triangleq (\mathsf{app}(t,t'), g(t,t',x,x')) \\ h'(a,(t,x)) &\triangleq (\mathsf{lam}(a.t), h(t,a,x)) \end{aligned}$$

Note that h' satisfies condition (66) because of property (38) and because h satisfies (68). So let $k' \in \Lambda_\alpha \to X'$ be the function uniquely defined by iteration from (f', g', h') as in Theorem 6.5. Writing π_1 and π_2 for the projection functions $(t,x) \mapsto t$ and $(t,x) \mapsto x$, one verifies that $\pi_1 \circ k'$ and the identity function $\mathsf{id}_{\Lambda_\alpha} \in \Lambda_\alpha \to \Lambda_\alpha$ both satisfy the conditions required of the function iteratively defined by $(\mathsf{var}, \mathsf{app}, \mathit{lam})$, where $\mathit{lam} \in (\mathsf{A} \times \Lambda_\alpha) \to \Lambda_\alpha$ is $(a,t) \mapsto \mathsf{lam}(a.x)$. (Note that this triple of functions is suitable for a structurally iterative definition because *lam* satisfies the necessary condition (66), thanks to property (38).) Hence $\pi_1 \circ k' = \mathsf{id}_{\Lambda_\alpha}$. Using this one easily verifies that $k = \pi_2 \circ k' \in \Lambda_\alpha \to X$ is the uniquely defined function with the required property (69). □

Theorem 6.8 (Structural induction for Λ_α). Given a subset $S \subseteq \Lambda_\alpha$ in $\mathcal{FM}(\mathbb{A})$, to prove that S is the whole of Λ_α it suffices to show

$$\begin{array}{ll} & \forall a \in \mathsf{A}(\mathsf{var}(a) \in S) \\ \wedge & \forall t \in S \forall t' \in S(\mathsf{app}(t,t') \in S) \\ \wedge & \mathit{И}a \in \mathsf{A} \forall t \in S(\mathsf{lam}(a.t) \in S) \end{array} \tag{70}$$

Proof. Condition (70) tells us that the inclusion

$$\mathsf{A} + (S \times S) + ([\mathsf{A}]S) \subseteq \mathsf{A} + (\Lambda_\alpha \times \Lambda_\alpha) + ([\mathsf{A}]\Lambda_\alpha) = \Lambda_\alpha$$

factors through $S \subseteq \Lambda_\alpha$ (we use property (46) to see this for the third component of the disjoint union). Hence $F_\alpha(S) \subseteq S$ and therefore by property (52) we have $\Lambda_\alpha = \mu(F_\alpha) \subseteq S$. Thus $S = \Lambda_\alpha$. □

This structural induction principle for Λ_α seems to correspond well to informal inductive arguments about α-equivalence classes of λ-terms that proceed by picking representative names of bound variables and applying structural induction at the level of abstract syntax trees (i.e. structural induction for $\Lambda = \mu(F)$, as defined in Theorem 6.2), leaving mute the tedious proofs that such choices do not affect the argument. In effect, by restricting to objects with finite support (Definition 3.3) and equivariant properties (Lemma 4.7), FM-set theory ensures that such choices are guaranteed not to affect meaning, provided freshness conditions like property (66) in Theorem 6.5 are satisfied.

In the rest of this section we give some simple examples of the use of Theorems 6.5 and 6.8.

Example 6.9 (Capture-avoiding substitution). Given $s \in \Lambda_\alpha$ and $b \in \mathsf{A}$, in Theorem 6.5 take $X = \Lambda_\alpha$ and define $f \in \mathsf{A} \to \Lambda_\alpha$, $g \in (\Lambda_\alpha \times \Lambda_\alpha) \to \Lambda_\alpha$ and $h \in (\mathsf{A} \times \Lambda_\alpha) \to \Lambda_\alpha$ by

$$\begin{aligned} f(a) &\triangleq \mathsf{if}\ a = b\ \mathsf{then}\ s\ \mathsf{else}\ \mathsf{var}(a) \\ g(t,t') &\triangleq \mathsf{app}(t,t') \\ h(a,t) &\triangleq \mathsf{lam}(a.t) \end{aligned}$$

Note that h satisfies condition (66) because of property (38). So by the theorem there is a unique function in $\Lambda_\alpha \to \Lambda_\alpha$, whose value at $t \in \Lambda_\alpha$ we write as $\mathsf{sub}\ s\ \mathsf{for}\ b\ \mathsf{in}\ t$, satisfying

$$\begin{aligned} & \forall a \in \mathsf{A} \quad \mathsf{sub}\ s\ \mathsf{for}\ b\ \mathsf{in}\ \mathsf{var}(a) = \\ & \qquad\qquad \mathsf{if}\ a = b\ \mathsf{then}\ s\ \mathsf{else}\ \mathsf{var}(a) \\ \wedge\ & \forall t' \in \Lambda_\alpha\, \forall t'' \in \Lambda_\alpha \quad \mathsf{sub}\ s\ \mathsf{for}\ b\ \mathsf{in}\ \mathsf{app}(t,t') = \\ & \qquad\qquad \mathsf{app}(\mathsf{sub}\ s\ \mathsf{for}\ b\ \mathsf{in}\ t, \mathsf{sub}\ s\ \mathsf{for}\ b\ \mathsf{in}\ t') \\ \wedge\ & И a \in \mathsf{A}\, \forall t \in \Lambda_\alpha \quad \mathsf{sub}\ s\ \mathsf{for}\ b\ \mathsf{in}\ \mathsf{lam}(a.t) = \\ & \qquad\qquad \mathsf{lam}(a.(\mathsf{sub}\ s\ \mathsf{for}\ b\ \mathsf{in}\ t)) \end{aligned}$$

From the above properties it follows that under the bijection of Theorem 6.2 between elements s of Λ_α and α-equivalence classes of λ-terms N, the function $\mathsf{sub}\ s\ \mathsf{for}\ b\ \mathsf{in}\ (-)$ corresponds to the capture-avoiding substitution function $[N/b](-)$. Note that in the clause for λ-abstractions $\mathsf{lam}(a.t)$, we only have to specify the result of substitution for the simple case that a is fresh in order for $\mathsf{sub}\ s\ \mathsf{for}\ b\ \mathsf{in}\ (-)$ to be defined on the whole of Λ_α.

One property of capture-free substitution is that the only free occurrences of b in $[N/b]M$ are due to free occurrences of b in N. The analogue for Λ_α of this property is

$$b \# s \ \Rightarrow\ \forall t \in \Lambda_\alpha (b \# \mathsf{sub}\ s\ \mathsf{for}\ b\ \mathsf{in}\ t) \tag{71}$$

This can be proved by structural induction on t by taking $S = \{t \in \Lambda_\alpha \mid b \# \mathsf{sub}\ s\ \mathsf{for}\ b\ \mathsf{in}\ t\}$ in Theorem 6.8 and making use of the easily verified fact that

$$b \# s \ \Rightarrow\ b \# (\mathsf{if}\ a = b\ \mathsf{then}\ s\ \mathsf{else}\ \mathsf{var}(a))$$

Applying Proposition 4.10 to (71), we have for each $s \in \Lambda_\alpha$ that

$$И b \in \mathsf{A}\, \forall t \in \Lambda_\alpha (b \# \mathsf{sub}\ s\ \mathsf{for}\ b\ \mathsf{in}\ t)$$

Thus by Lemma 6.3, for each $s \in \Lambda_\alpha$ the element of $(\mathsf{A} \times \Lambda_\alpha) \to \Lambda_\alpha$ given by $(b,t) \mapsto \mathsf{sub}\ s\ \mathsf{for}\ b\ \mathsf{in}\ t$ induces a function $([\mathsf{A}]\Lambda_\alpha) \to \Lambda_\alpha$. In this way we get a substitution function $\sigma \in (([\mathsf{A}]\Lambda_\alpha) \times \Lambda_\alpha) \to \Lambda_\alpha$ satisfying for all $(y,s) \in ([\mathsf{A}]\Lambda_\alpha) \times \Lambda_\alpha$ that $И a \in \mathsf{A}(\sigma(y,s) = \mathsf{sub}\ s\ \mathsf{for}\ a\ \mathsf{in}\ (y@a))$. (Compare with the substitution function $\sigma : \delta\Lambda \times \Lambda \to \Lambda$ in [FPT99, Section 3].)

Example 6.10 (A size function). In Theorem 6.5, taking X to be the set ω of natural numbers, $f \in \mathsf{A} \to \omega$ to be the constant function $a \mapsto 1$, $g \in (\omega \times \omega) \to \omega$ to be addition and $h \in (\mathsf{A} \times \omega) \to \omega$ to be the function $(a,n) \mapsto n+1$, we can deduce that there is a unique function $k \in \Lambda_\alpha \to \omega$ satisfying

$$\begin{aligned} & \forall a \in \mathsf{A}(k(\mathsf{var}(a)) = 1) \\ \wedge\ & \forall t \in \Lambda_\alpha\, \forall t' \in \Lambda_\alpha (k(\mathsf{app}(t,t')) = k(t) + k(t')) \\ \wedge\ & И a \in \mathsf{A}\, \forall t \in \Lambda_\alpha (k(\mathsf{lam}(a.t)) = k(t) + 1) \end{aligned}$$

Condition (66) is satisfied in this case because ω is a pure FM-set (see Remark 4.2) and hence $\forall a \in \mathsf{A}\, \forall n \in \omega (a \# n)$.

By Proposition 4.10, the last property of k in the above conjunction is equivalent to

$$\forall a \in \mathsf{A}(a \# k \;\Rightarrow\; \forall t \in \Lambda_\alpha (k(\mathsf{lam}(a.t)) = k(t) + 1))$$

But Corollary 4.8 implies that $a \# k$ holds for any $a \in \mathsf{A}$. Therefore we can strengthen this last defining clause for k to

$$\forall a \in \mathsf{A}\, \forall t \in \Lambda_\alpha (k(\mathsf{lam}(a.t)) = k(t) + 1))$$

Thus the formalism we have developed allows us to express very easily the properties we expect a size function to have on α-equivalence classes of λ-terms. Compare this example with the complications encountered by Gordon and Melham when defining a similar function by recursion using their axiomatisation of α-conversion [GoM96, Section 3.3].

Example 6.11 (Set of free atoms). In Theorem 6.5, take X to be the set $\mathsf{pow}_{\mathrm{fin}}(\mathsf{A})$ of finite sets of atoms, $f \in \mathsf{A} \to \mathsf{pow}_{\mathrm{fin}}(\mathsf{A})$ to be the singleton function $a \mapsto \{a\}$, $g \in (\mathsf{pow}_{\mathrm{fin}}(\mathsf{A}) \times \mathsf{pow}_{\mathrm{fin}}(\mathsf{A})) \to \mathsf{pow}_{\mathrm{fin}}(\mathsf{A})$ to be union and $g \in (\mathsf{A} \times \mathsf{pow}_{\mathrm{fin}}(\mathsf{A})) \to \mathsf{pow}_{\mathrm{fin}}(\mathsf{A})$ to be the subtraction function $(a, w) \mapsto w - \{a\}$. Note that h satisfies condition (66), because the support of a finite set of atoms $w \in \mathsf{pow}_{\mathrm{fin}}(\mathsf{A})$ is just w itself and hence $a \# (w - \{a\})$, for any $a \in \mathsf{A}$. So there is a unique function $k \in \Lambda_\alpha \to \mathsf{pow}_{\mathrm{fin}}(\mathsf{A})$ satisfying

$$\begin{array}{ll} & \forall a \in \mathsf{A}(k(\mathsf{var}(a)) = \{a\}) \\ \wedge & \forall t \in \Lambda_\alpha\, \forall t' \in \Lambda_\alpha (k(\mathsf{app}(t, t')) = k(t) \cup k(t')) \\ \wedge & \forall a \in \mathsf{A}\, \forall t \in \Lambda_\alpha (k(\mathsf{lam}(a.t)) = k(t) - \{a\}) \end{array}$$

where in the last conjunct we have strengthened $И a \in \mathsf{A} \ldots$ to $\forall a \in \mathsf{A} \ldots$ using the same argument as in the previous example. This function k gives a structurally iterative definition of the 'finite set of free atoms' for elements of Λ_α. Using the structural induction principle of Theorem 6.8, one can show for all $t \in \Lambda_\alpha$ that in fact $k(t) = \mathsf{supp}(t)$. Thus under the bijection of Theorem 6.2, the set of (names of) free variables of an α-equivalence class of λ-terms is identified with the support of the corresponding element of Λ_α (cf. Example 3.5(iii)). In particular, $\{t \in \Lambda_\alpha \mid \mathsf{supp}(t) = \emptyset\}$ corresponds to the subset of *closed* λ-terms modulo $=_\alpha$.

Example 6.12 (Binding occurrences of atoms). It is simple enough to define a function $\Lambda \to \mathsf{pow}_{\mathrm{fin}}(\mathsf{A})$ sending each λ-term M to the finite set of atoms that occur in M in binding position. Clearly this set is not invariant under α-conversion and so does not induce a function on $\Lambda/{=_\alpha}$. Correspondingly, there is no function $ba \in \Lambda_\alpha \to \mathsf{pow}_{\mathrm{fin}}(\mathsf{A})$ in $\mathcal{FM}(\mathbb{A})$ picking out the finite set of 'binding atoms' of elements of Λ_α, in the sense that it satisfies

$$\begin{array}{ll} & \forall a \in \mathsf{A}(ba(\mathsf{var}(a)) = \emptyset) \\ \wedge & \forall t \in \Lambda_\alpha\, \forall t' \in \Lambda_\alpha (ba(\mathsf{app}(t, t')) = ba(t) \cup ba(t')) \\ \wedge & И a \in \mathsf{A}\, \forall t \in \Lambda_\alpha (ba(\mathsf{lam}(a.t)) = \{a\} \cup ba(t)) \end{array}$$

(Note that we cannot use Theorem 6.5 to define such a function because in this case condition (66) would require $a \# \{a\} \cup w$ (any $w \in \mathsf{pow}_{\mathrm{fin}}(\mathsf{A})$), which is certainly false.) Indeed, we can argue by contradiction to see that no such function exists in $\mathcal{FM}(\mathbb{A})$: if it did, picking any atoms $a \neq a'$ not in its support, we would have $(a'\ a) \cdot ba = ba$ (by property (15)); then $t = \mathsf{lam}(a.\mathsf{var}(a))$ satisfies $ba(t) = \{a\}$ and so

$$\begin{array}{rll} \{a'\} = (a'\ a) \cdot \{a\} & = (a'\ a) \cdot ba(t) & \\ & = ba(t) & \text{since } a, a' \# ba(t), \text{ by (24)} \\ & = \{a\} & \end{array}$$

i.e. $a' = a$, contradicting the choice of a and a'.

The theory we have presented seems rather well adapted to expressing the 'usual' forms of recursion/induction for abstract syntax, while at the same time dealing with freshness of variables and variable renaming systematically, at the meta-level. Of course much more needs to be done to establish the utility of these structural recursion and induction principles (see Section 8). However, we regard the sheer simplicity of the above examples (compared with analogous examples in other formalisms) as a good sign! We finish this section by mentioning some extensions of the theory.

Remark 6.13 (Binding signatures). The terms of the lambda calculus are the terms over a signature with one binary operation (application) and one unary binder (λ-abstraction). The results of this section can be extended to deal with more general signatures, consisting of k operators (for some $k \geq 0$) each of which takes m arguments (for some $m \geq 0$) with each argument an n-ary abstraction (for some $n \geq 0$) — these are the *binding signatures* considered in [FPT99, Section 2]. The function F on FM-sets corresponding to such a signature takes the form

$$F(-) = \mathsf{A} + \overbrace{\cdots + (\underbrace{\cdots \times [\mathsf{A}]^n(-) \times \cdots}_{m \text{ factors}}) + \cdots}^{k \text{ summands}}$$

where

$$\begin{cases} [\mathsf{A}]^0(X) & \triangleq X \\ [\mathsf{A}]^{n+1}(X) & \triangleq [\mathsf{A}]([\mathsf{A}]^n(X)) \end{cases}$$

Generalising Theorem 6.2, one can show that the inductively defined FM-set $\mu(F)$ determined by F is in bijection with the set of terms modulo α-conversion over the binding signature. This FM-set is an initial algebra for the functor associated with F; and from the shape of the signature can be read off principles of structural recursion and induction like those above. Many-sorted signatures can be dealt with using FM-sets mutually inductively defined by several such functions. See [Gab00, Section 10] for more details.

Remark 6.14 (Splitting the set of atoms). To support syntax involving finitely many *different sorts of variables* one can use the following mild generalisation of the Fraenkel-Mostowski universe $\mathcal{FM}(\mathbb{A})$ of Definition 4.1. Partition the set of atoms $\mathbb{A}$ into a countably infinite set of infinite subsets:

$$\begin{array}{ll} & \mathbb{A} = \bigcup\{A(n) \mid n \in \omega\} \\ \wedge & \forall n \in \omega(A(n) \notin \mathsf{pow}_{\mathrm{fin}}(A(n))) \\ \wedge & \forall n, n' \in \omega(n \neq n' \Rightarrow A(n) \cap A(n') = \emptyset) \end{array}$$

Working with sets equipped with an action of the subgroup G of $perm(\mathbb{A})$ consisting of permutations that respect the partition (so that G is isomorphic to the product group $\prod_{n\in\omega} perm(A(n))$), we can repeat the development of Sections 3 and 4, forming a universe of sets with hereditarily finite support with respect to the action of G. There is a corresponding generalisation of FM-set theory to *Fraenkel-Mostowski set theory with many Sorts of atoms* (FMS-set theory), whose axioms we will not give here. Within that theory one can develop the theory of transpositions $(a\,b) \cdot_A x$, a freshness predicate $a \#_A x$, a quantifier $\mathcal{N}a \in A(-)$ and abstraction sets $[A]X$, where $a, b \in A \in \{A(n) \mid n \in \omega\}$. The latter can be used in combination with Cartesian products and disjoint unions to form inductively defined sets in the universe just as in Section 6. For example, taking two distinct sorts of atoms, $T \triangleq A(0)$ and $V \triangleq A(1)$ naming type variables and term variables respectively, the mutually inductively defined FMS-sets

$$\left.\begin{array}{rcl} \mathit{Type} & = & \mathsf{tyvar}(T) + \mathsf{Fun}(\mathit{Type} \times \mathit{Type}) + \mathsf{all}([T]\,\mathit{Type}) \\ \mathit{Term} & = & \mathsf{var}(V) + \mathsf{app}(\mathit{Term} \times \mathit{Term}) + \mathsf{lam}(\mathit{Type} \times [V]\mathit{Term}) \\ & & \quad + \mathsf{spec}(\mathit{Term} \times \mathit{Type}) + \mathsf{gen}([T]\,\mathit{Term}) \end{array}\right\} \tag{72}$$

represent, respectively, the types and terms of the Girard-Reynolds polymorphic lambda calculus [Gir72, Rey83] *modulo renaming of bound type variables and bound term variables.* (The disjoint union inclusion functions have been named explicitly in (72) to indicate the intended meaning.)

7. Related Work

One origin of the work presented here lies in the 'ν-calculus', a fragment of ML [MTHM97] introduced by the second author and Stark [PiS93a, Sta95] to explore the properties, with respect to semantic equivalence of programs, of call-by-value higher order functions and dynamically created names (see also [JeR99]). In [Sta96a], Stark studies a model of the ν-calculus based on one of Moggi's 'dynamic allocation' monads [Mog89] in the presheaf category $\mathbf{Set}^{\mathcal{I}}$, where $\mathcal{I}$ is the category of finite ordinals and injective functions between them. Crucial ingredients of the dynamic allocation monad used there are the 'object of names',

given by the inclusion functor $\mathcal{I} \hookrightarrow \mathbf{Set}$, and the shift functor $\delta : \mathbf{Set}^{\mathcal{I}} \to \mathbf{Set}^{\mathcal{I}}$, given by $\delta X(n) = X(n+1)$. These ingredients also occur in subsequent work on modelling π-calculus names in $\mathbf{Set}^{\mathcal{I}}$ [FMS96, Sta96b] and, most relevantly, recent work on modelling variable-binding abstract syntax [FPT99, Hof99], where other presheaf categories besides $\mathbf{Set}^{\mathcal{I}}$ are considered.

Now, a somewhat overlooked model of the ν-calculus, mentioned in [PiS93b, Examples 4.3], is the full subcategory of $\mathbf{Set}^{\mathcal{I}}$ whose objects are the pullback preserving functors. This is equivalent to a well-known topos, sometimes called the *Schanuel topos* — the category of continuous G-sets for the topological group $G = perm(\mathbb{A})$ of permutations of a countably infinite set $\mathbb{A}$ topologised as a subspace of Baire space: see [Joh83, Lemma 1.8] and [MaM92, Section III.9]. Put more concretely, and this is the point, the objects of the Schanuel topos are $perm(\mathbb{A})$-sets (Definition 3.1) in which every element has finite support (Definition 3.3), and its morphisms are the equivariant functions defined at the beginning of Section 4. Thus on the one hand, the Schanuel topos relates to the cumulative hierarchy of FM-sets defined by equation (11) much as the usual von Neumann cumulative hierarchy, defined by equation (9), relates to the topos of sets (see [JoM95] for more on the category theory of universes of sets); on the other hand, the Schanuel topos is a sheaf subtopos of the presheaf category $\mathbf{Set}^{\mathcal{I}}$, with the inclusion sending the FM-set of atoms $\mathbb{A}$ to the object of names $\mathcal{I} \hookrightarrow \mathbf{Set}$ and the abstraction operator $[\mathbb{A}](-)$ to the shift functor $\delta(-)$ mentioned above.

Both the presheaf toposes used in [FPT99, Hof99] and the Schanuel topos (and indeed many other categories equipped with a faithful functor to $\mathbf{Set}^{\omega}$) support an initial algebra semantics for signatures with binding. So does the Schanuel topos, and its associated FM-set theory, have any advantage over these other, related categories? It is well known that toposes correspond to theories in extensional, higher-order, intuitionistic logic [LaS86]. Unlike presheaf toposes in general, the Schanuel topos models *classical* rather than intuitionistic higher-order logic; furthermore, its higher-order structure (function and power objects) is rather amenable to making calculations, compared with the higher-order structure of presheaf categories. Thus if one is looking for a single, general-purpose setting for modelling variable-binding syntax, the logic of the Schanuel topos is both a bit more powerful and familiar. One can view [FPT99] as establishing, amongst other things, a very nice categorical algebra for the de Buijn, 'nameless' style of treating variable-binding and substitution. By contrast, here we have presented a useful logic for variable-binding in which names occur explicitly. The motivation for such a 'nameful' rather than nameless style is partly to formalise existing common practice; but the logic can also serve as a basis for proving that de Bruijn-like formulations are correct with respect to more concrete representations. Crucial to all this is the notion of 'finite support' (leading to the N-quantifier and our FM-set-theoretic notion of atom-abstraction), which is present in the Schanuel toposes, but not, apparently, in the presheaf toposes used in [FPT99] — or at least not with such pleasant properties.

Finally, it is worth emphasising that although we have chosen to use a set-theoretic rather than a topos-theoretic presentation here — because we think it is more accessible — the same fundamental ideas to do with names and name-abstraction underlie both FM-set theory and the Schanuel topos.

8. Conclusion and Further Developments

The Fraenkel-Mostowski permutation model of ZFA is well over sixty years old. Yet the use to which we have put it here seems new. The idea that one might want to treat syntax up to permutative renamings of variables is hardly original, but we hope we have demonstrated convincingly that the theory really takes off when one combines that idea with the subtle notion of 'finite support' inherent in the Fraenkel-Mostowski model. Using it, we introduced a useful quantifier N for fresh names and a new set-forming operation for name-abstraction, $[\mathsf{A}](-)$, whose properties seem better suited to modelling variable-binding than the function space $\mathsf{A} \to (-)$ that is often used for that purpose. Among other things, we saw that the theory of inductively defined FM-sets using this notion of abstraction can correctly model α-equivalence classes of variable-binding syntax. At the same time we remain pleasantly close to the familiar theory of first-order algebraic data types — witness Theorems 6.5 and 6.8, and Examples 6.9–6.12.

We believe that the theory of inductively defined FM-sets will be a useful setting for developing programming language semantics based on structural operational semantics [Plo81]. Syntax-directed, rule-based inductive definitions of relations quite often contain side-conditions to do with freshness of variables, and the hope is that these can be assimilated and manipulated conveniently via the freshness predicate # and the N-quantifier we have introduced here. This approach seems particularly relevant for the labelled transi-

tion semantics of process calculi such as the π-calculus and its relations [MPW92, AbG99, CaG00b]; both because complicated forms of binding to do with names and names restriction play a key role there and because analyses based on 'name-abstraction as functions' [HMS98, Des00] are not entirely satisfactory. The formalism of permutation actions and finite support has been rediscovered independently by Honda in his work on a general framework for processes [Hon00]. It also features in work on finite state transition systems for checking bisimilarity of π-calculus processes in [MoP00]. Cardelli and Gordon have taken up some of the ideas presented here and used permutative renaming and the $\mathcal{N}$-quantifier as part of a modal logic for their ambient calculus [CaG00a].

This paper is a revised and expanded version of [GaP99]. Since that paper appeared, the theory we have described here has been developed and applied in two somewhat different directions. The first author has developed an implementation of FM-set theory within the Isabelle generic theorem prover [Pau94]. This required a good deal of 'proof engineering' to reuse the existing Isabelle ZF theory; an account may be found in Chapter III of the first author's thesis [Gab00] (Chapter II of which contains a development of FM-set theory more detailed than the one given here). Secondly, we have begun to design an ML-style metalanguage, `FreshML`, for programming with recursively defined functions on inductively defined FM-sets. The ingredients of `FreshML` are described in [PiG00]. It has a type of atoms, a type constructor for atom-abstractions, and facilities for defining recursive functions involving locally fresh names and pattern-matching on atom-abstractions (cf. Remark 6.4). The `FreshML` type system infers not only conventional typing information, but also information about freshness assertions $a \# x$; the crucial point is that the type system ensures that the language's facilities for manipulating bound names can only be used in well-typed programs in ways that are insensitive to renaming those bound names.

References

[AbG99] Abadi, M. and Gordon, A. D.: A calculus for cryptographic protocols: the spi calculus. *Information and Computation*, 148:1–70, 1999.

[Bar84] Barendregt, H. P.: *The Lambda Calculus: Its Syntax and Semantics* (rev. edn). North-Holland, Amsterdam, 1984.

[Bur69] Burstall R. M.: Proving properties of programs by structural induction. *Computer Journal*, 12:41–48, 1969.

[Bur77] Burstall, R. M.: Design considerations for a functional programming language. In *Proc. of the Infotech State of the Art Conference, Copenhagen*, 1977.

[CaG00a] Cardelli, L. and Gordon A. D.: Logical properties of name restriction. In *Typed Lambda Calculus and Applications, 5th International Conference*, Vol. 2044 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, 2001.

[CaG00b] Cardelli, L. and Gordon, A. D.: Mobile ambients. *Theoretical Computer Science*, 240:177–213, 2000.

[Chu40] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[Cur93] P.-L. Curien. *Categorical Combinators, Sequential Algorithms, and Functional Programming.* Birkhäuser, Basel, 1993.

[dB72] de Bruijn, N. G.: Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indag. Math.*, 34:381–392, 1972.

[Des00] Despeyroux, J.: A higher-order specification of the pi-calculus. In *IFIP International Conference on Theoretical Computer Science, IFIP TCS'2000*, Sendai, Japan, 2000.

[DPS97] J. Despeyroux, F. Pfenning, and C. Schürmann. Primitive recursion for higher-order abstract syntax. In *Typed Lambda Calculus and Applications, 3rd International Conference*, Vol.1210 of Lecture Notes in Computer Science. Springer, Berlin, 1997, pp. 147–163.

[FMS96] Fiore, M. P., Moggi, E. and Sangiorgi, D.: A fully abstract model for the π-calculus (extended abstract). In *Eleventh Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, Washington, DC, 1996, pp. 43–54.

[Fou80] Fourman, M. P.:. Sheaf models for set theory. *Journal of Pure and Applied Algebra*, 19:91–101, 1980.

[FPT99] Fiore, M. P., Plotkin, G. D. and Turi, D.: Abstract syntax and variable binding. In *14th Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, Washington, DC, 1999, pp. 193–202

[Gab00] Gabbay, M. J.: *A Theory of Inductive Definitions with α-Equivalence: Semantics, Implementation, Programming Language*. PhD thesis, Cambridge University, 2000.

[Gir72] Girard, J.-Y.: *Interprétation fonctionelle et élimination des coupures dans l'arithmetique d'ordre supérieur*. PhD thesis, Université Paris VII, 1972.

[GoM93] Gordon M. J. C. and Melham T. F.: *Introduction to HOL. A theorem proving environment for higher order logic.* Cambridge University Press, Cambridge UK, 1993.

[GoM96] Gordon A. D. and Melham, T.: Five axioms of alpha-conversion. In *Theorem Proving in Higher Order Logics: 9th Interational Conference, TPHOLs'96*, Vol. 1125 of Lecture Notes in Computer Science. Springer, Berlin, 1996, pp. 173–191

[GaP99] Gabbay, M. J. and Pitts, A. M.: A new approach to abstract syntax involving binders. In *14th Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, Washington, DC, 1999, pp. 214–224.

[GTWW77] Goguen, J. A. , Thatcher, J. W., Wagner, E. G. and Wright, J. B.: Initial algebra semantics and continuous algebras. *Journal of the Association for Computing Machinery*, 24:68–95, 1977.

[Gun92] Gunter, C. A.: *Semantics of Programming Languages: Structures and Techniques.* Foundations of Computing. MIT Press, Cambridge, MA, 1992.

[HMS98] Honsell, F., Miculan, M. and Scagnetto, I.: π-Calculus in (co)inductive type theory. Technical report, Dipartimento di Matematica e Informatica, Università degli Studi di Udine, 1998.

[Hof99] Hofmann, M.: Semantical analysis of higher-order abstract syntax. In *14th Annual Symposium on Logic in Computer Science.* IEEE Computer Society Press, Washington, DC, 1999, pp. 204–213.

[Hon00] Honda, K.: Elementary structures in process theory (1): Sets with renaming. *Mathematical Structures in Computer Science*, 10:617–663, 2000.

[Jec77] Jech, T. J.: About the axiom of choice. In J. Barwise, editor, *Handbook of Mathematical Logic.* North-Holland, Amsterdam, 1977, pp. 345–370.

[JoM95] Joyal, A. and Moerdijk, I.: *Algebraic Set Theory.* Number 220 in London Mathematical Society Lecture Notes in Mathematics. Cambridge University Press, Cambridge, UK, 1995.

[Joh83] Johnstone, P. T.: Quotients of decidable objects in a topos. *Math. Proc. Cambridge Philosophical Society*, 93:409–419, 1983.

[JeR99] Jeffrey, A. and Rathke, J.: Towards a theory of bisimulation for local names. In *14th Annual Symposium on Logic in Computer Science.* IEEE Computer Society Press, Washington, DC, 1999, , pp. 56–66.

[LaP99] Lamport, L. and Paulson, L. C.: Should your specification language be typed? *ACM Transactions on Programming Languages and Systems*, 21(3):502–526, 1999.

[LaS86] Lambek, J. and Scott, P. J.: *Introduction to Higher Order Categorical Logic.* Cambridge University Press, Cambridge, UK, 1986.

[Mar84] Martin-Löf, P.: *Intuitionistic Type Theory.* Bibliopolis, Naples, 1984.

[MaM92] MacLane, S. and Moerdijk, I.: *Sheaves in Geometry and Logic. A First Introduction to Topos Theory.* Springer, New York, 1992.

[McM97] McDowell, R. and Miller, D: A logic for reasoning with higher-order abstract syntax. In *12th Annual Symposium on Logic in Computer Science.* IEEE Computer Society Press, Washington, DC, 1997, , pp. 434–445.

[Mog89] Moggi, E.: An abstract view of programming languages. Technical Report ECS-LFCS-90-113, Department of Computer Science, University of Edinburgh, 1989.

[MoP00] Montanari, U. and Pistore, M.: π-Calculus, structured coalgebras and minimal HD-automata. In *25th International Symposium on Mathematical Foundations of Computer Science*, Bratislava, Slovak Republic, Vol. 1893 of Lecture Notes in Computer Science. Springer, Berlin, 2000.

[MPW92] Milner, M., Parrow, J. and Walker, D.: A calculus of mobile processes (parts I and II). *Information and Computation*, 100:1–77, 1992.

[MTHM97] Milner, R., Tofte, M., Harper, R. and MacQueen, D.: *The Definition of Standard ML (Revised).* MIT Press, Cambridge, MA, 1997.

[Pau94] Paulson, L. C.: *Isabelle: A Generic Theorem Prover.* Vol. 828 of *Lecture Notes in Computer Science.* Springer, Berlin, 1994.

[PfE88] Pfenning, F. and Elliott, C.: Higher-order abstract syntax. In *Proc. ACM-SIGPLAN Conference on Programming Language Design and Implementation.* ACM Press, New York, 1988, pp. 199–208.

[PiG00] Pitts, A. M. and Gabbay, M. J.: A metalanguage for programming with bound names modulo renaming. In R. Backhouse and J. N. Oliveira, editors, *Mathematics of Program Construction. 5th International Conference, MPC2000*, Ponte de Lima, Portugal, July 2000. Vol. 1837 of Lecture Notes in Computer Science. Springer, Heidelberg, 2000, pp. 230–255.

[Plo81] Plotkin, G. D.: A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University, 1981.

[PaM93] Paulin-Mohring, C.: Inductive definitions in the system Coq; rules and properties. In M. Bezem and J. F. Groote, editors, *Typed Lambda Calculus and Applications.* Vol. 664 of Lecture Notes in Computer Science. Springer, Berlin, 1993, pp. 328–345.

[PiS93a] Pitts, A. M. and Stark, I. D. B. : Observable properties of higher order functions that dynamically create local names, or: What's new? In *Mathematical Foundations of Computer Science, Proceedings of the 18th International Symposium*, Gdańsk. Vol. 711 of Lecture Notes in Computer Science. Springer, Berlin, 1993, pp. 122–141.

[PiS93b] Pitts, A. M. and Stark, I. D. B. : On the observable properties of higher order functions that dynamically create local names (preliminary report). In *Workshop on State in Programming Languages*, Copenhagen. ACM SIGPLAN. Technical Report YALEU/DCS/RR-968, Yale University Department of Computer Science, 1993, pp. 31–45.

[Qui63] Quine, W. V. O.: *Set Theory and its Logic* (rev. edn). Harvard University Press, Cambridsge, MA, 1963.

[Rey83] Reynolds, J. C.: Types, abstraction and parametric polymorphism. In R. E. A. Mason, editor, *Information Processing 83.* North-Holland, Amsterdam, 1983, pp. 513–523.

[Sco93] Scott, D. S.: A type-theoretical alternative to ISWIM, CUCH, OWHY. *Theoretical Computer Science*, 121:411–440, 1993.

[Sho77] Shoenfield, J. R.: Axioms of set theory. In J. Barwise, editor, *Handbook of Mathematical Logic.* North-Holland, Amsterdam, 1977, pp. 321–344.

[Sta95] Stark, I. D. B.: *Names and Higher-Order Functions.* PhD thesis, University of Cambridge, 1995. Also published as Technical Report 363, University of Cambridge Computer Laboratory, April 1995.

[Sta96a] Stark, I. D. B.: Categorical models for local names. *Lisp and Symbolic Computation*, 9(1):77–107, 1996.

[Sta96b] Stark, I. D. B.: A fully abstract domain model for the π-calculus. In *11th Annual Symposium on Logic in Computer Science.* IEEE Computer Society Press, Washington, DC, 1996, pp. 36–42.

[Sto88] Stoughton, A.: Substitution revisited. *Theoretical Computer Science*, 59:317–325, 1988.